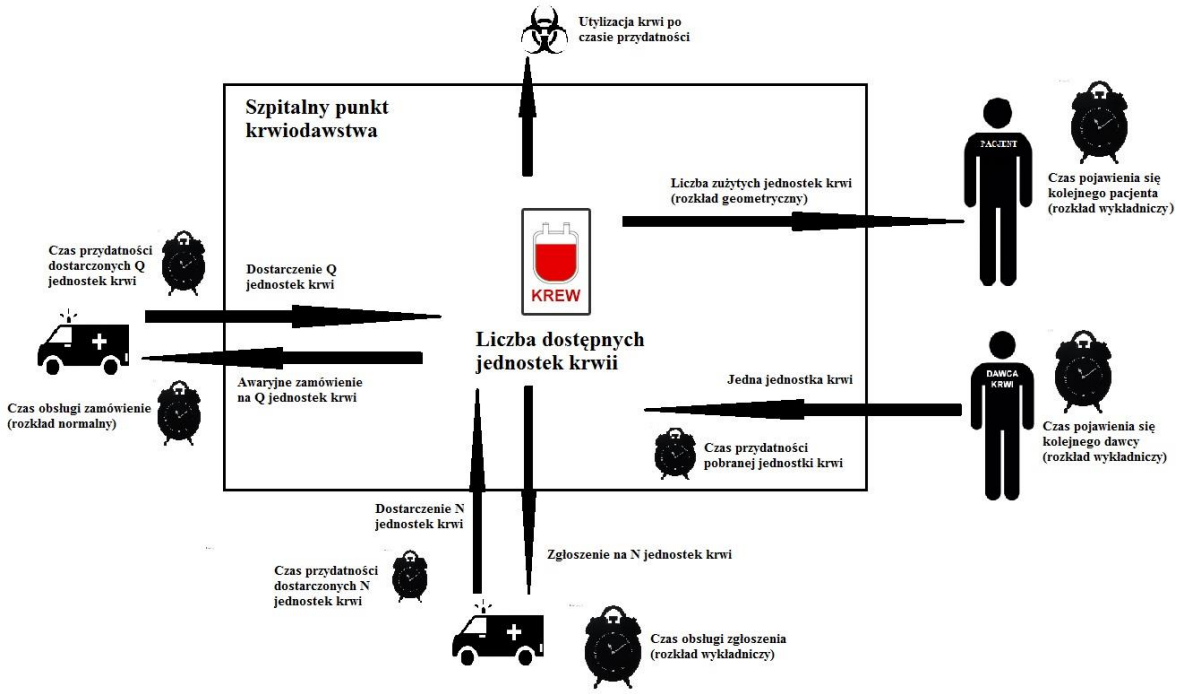
RAPORT SYMULACJA CYFROWA

1. **Nr rozszerzenia zadania A1, tabela D3, metoda planowania zdarzeń.**
2. **Treść rozwiązywanego zadania:**
   Szpitalny punkt krwiodawstwa korzysta z monitoringu liczby dostępnych jednostek krwi. Jeżeli liczba ta spadnie do poziomu **R** lub niżej, zostaje wysłane zlecenie na **N** nowych jednostek. Czas od wysłania zamówienia do otrzymania krwi jest zmienną losową o rozkładzie wykładniczym o średniej **Z**. Dostarczona krew musi zostać wykorzystana w ciągu $\mathbf{T_1}$ jednostek czasu. Po tym czasie zostaje zutylizowana.
   Odstęp czasu pomiędzy pojawieniem się kolejnych pacjentów wymagających transfuzji jest zmienną losową o rozkładzie wykładniczym i średniej **P**. Liczba jednostek krwi podawana pojedynczemu pacjentowi jest zmienną losową o rozkładzie geometrycznym i średniej **1/W**. Jeżeli liczba potrzebnych jednostek jest większa niż aktualny stan zaopatrzenia w punkcie krwiodawstwa, zostaje złożone awaryjne zamówienie na **Q** jednostek. Czas dostarczenia takiego zamówienia jest zmienną losową o rozkładzie normalnym, średniej **E** i wariancji $\mathbf{EW^2}$. Dodatkowo, w punkcie krwiodawstwa krew oddają lokalni dawcy. Czas między zgłoszeniem się kolejnych dawców jest zmienną losową o rozkładzie wykładniczym i średniej **L**. Każdy dawca oddaje jedną jednostkę krwi, która musi zostać zużyta w ciągu $\mathbf{T_2}$ jednostek czasu ($\mathbf{T_1 < T_2}$).
   Punkt krwiodawstwa raz na TA jednostek czasu przeprowadza akcję promocyjną zbiórki krwi, podczas której średni czas pomiędzy zgłoszeniami kolejnych dawców spada o TR. Akcja trwa TT=7200 jednostek czasu.
   TA – zmienna losowa o rozkładzie równomiernym w przedziale [TAmin,TAmax] [20k,22k]
   TR – zmienna losowa o rozkładzie równomiernym w przedziale [TRmin,TRmax] [100,200]
   Celem symulacji jest wyznaczenie wartości **R** oraz **N,** dla których prawdopodobieństwo awaryjnego zamówienia jest mniejsze niż **A**. Dla otrzymanych wartości wyznacz jaki procent krwi jest utylizowany

3. **Opis modelu symulacyjnego:**

a) Schemat modelu symulacyjnego:



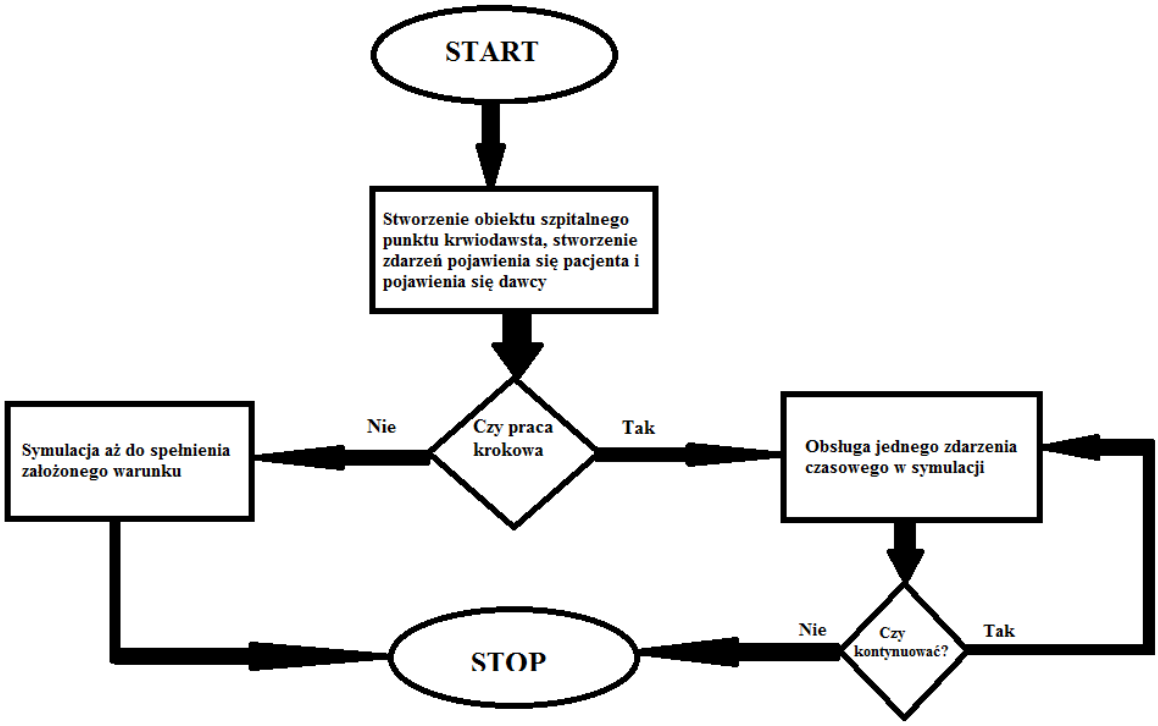
b) Opis klas wchodzących w skład systemu i ich atrybutów:

Obiekt	Nazwa klasy implementującej obiekt	Opis	Atrybuty
Pojawienie się dawcy krwi	ApperanceDonor	Klasa reprezentująca pojawienie się dawcy krwi	- Wskaźnik na szpitalny punkt krwiodawstwa typu *BloodDonationPoint* - Wskaźnik na obiekt generatora typu *Generators* - Zmienna wynikowa z generatora czasu typu *double*
Pojawienie się pacjenta	ApperancePatient	Klasa reprezentująca pojawienie się pacjenta	- Wskaźnik na szpitalny punkt krwiodawstwa typu *BloodDonationPoint* - Wskaźnik na obiekt generatora typu *Generators*
Pojawienie się krwi z awaryjnego zgłoszenia lub zamówienia	BloodDelivery	Klasa reprezentująca przywiezienie krwi	- Wskaźnik na szpitalny punkt krwiodawstwa typu *BloodDonationPoint* - Wskaźnik na obiekt generatora typu *Generators* - Zmienna wyliczeniowa mówiąca które zamówienie się pojawiło typu *TypeDelivery*
Szpitalny punkt krwiodawstwa	BloodDonationPoint	Klasa gromadząca pozostałe elementy sytemu.	- Minimalna liczba dostępnych jednostek krwi typu *const int* - Liczba krwi przychodząca z awaryjnego zgłoszenia i zamówienia typu *const int* - Liczba dostępnych jednostek krwi typu *int* - Flagi mówiące o tym czy zostało wysłane awaryjne zgłoszenie/zamówienie typu *bool* - Aktualny czas systemu typu *double*

			- Zmienne pozwalające na zakończenie symulacji (z zależności od parametrów końca) typu *int/double* - Zmienne do zbierania statysyk typu *int/double* - Zmienna do ustalania czy praca krokowa czy ciągła typu *bool* - Zmienna mówiąca czy trwa akcja promocyjna typu *bool* - Kolejka pacjentów typu *queue* - Lista krwi typu *list* - Kalendarz zdarzeń typu *list*
Utylizacja krwi	BloodRecycling	Klasa reprezentująca przeterminowanie krwi	- Wskaźnik na szpitalny punkt krwiodawstwa typu *BloodDonationPoint* - Wskaźnik na obiekt generatora typu *Generators*
Klasa wyliczeniowa	Enum	Klasa zawierająca typy wyliczeniowe	- Rodzaj krwi (czy od dawcy czy z dostawy) typu *enum* - Rodzaj dostawy typu *enum* - Rodzaj generatora typu *enum* - Czy start czy stop kampanii promocyjnej typu *enum* - Wartości do generatora typu *enum*
Zdarzenie czasowe	Event	Klasa nadrzędna dla zdarzeń czasowych	- Zmienna reprezentująca wykonania zdarzenia typu *double* - Czas przydatności krwi od dawcy typu *const int* - Czas przydatności krwi z dostawy typu *const int*
Generatory	Generators	Klasa zawierająca wszystkie generatory	- Zmienne typu *int* i *double* służące do generowania odpowiednich zmiennych o danym rozkładzie
Pacjent wymagający transfuzji	Patient	Klasa reprezentująca pacjenta wymagającego transfuzji krwi.	- Zużyte jednostki krwi przez pacjenta typu *int*
Kampania promocyjna	PromotionalCampainf	Klasa reprezentująca zdarzenie kampanii promocyjnej	- Wskaźnik na szpitalny punkt krwiodawstwa typu *BloodDonationPoint* - Wskaźnik na obiekt generatora typu *Generators* - Zmienna reprezentująca czas końca kampanii promocyjnej typu *const int* - Zmienna wyliczeniowa mówiąca o tym czy zdarzenie jest początkiem czy końcem kampanii promocyjnej typu *TypeCampaing*
Jednostka Krwi	UnitOfBlood	Klasa reprezentująca jednostkę krwi. Krew jest dodawana po zdarzeniach jak i utylizowana po	- czas przydatności krwi od dawcy lub z zamówienia typu *const double* - Zmienna reprezentująca czas do kiedy będzie przydatna krew typu *double*

		czasie jej przydatności	

4. **Opis przydzielonej metody symulacyjnej:**
   a) Schemat blokowy pętli głównej:



b) Lista zdarzeń czasowych/warunkowych:
Zdarzenia czasowe:

Zdarzenie	Opis	Algorytm
Pojawienie się pacjenta wymagającego transfuzji	Zdarzenie generowane jest w momencie pojawienia się pacjenta wymagającego transfuzji krwi. Pojawienie się kolejnego pacjenta wymagającego transfuzji krwi jest zmienną losową o rozkładzie wykładniczym i średniej P. Liczba jednostek krwi jaką zużywa pacjent jest zmienną losową o rozkładzie geometrycznym i średniej 1/W.	1. Generuj liczbę jednostek krwi jaką zużyje pacjent. 2. Sprawdź czy jest ktoś w kolejce pacjentów. a) Jeśli nie to: obsłuż zdarzenie warunkowe „Pobranie krwi przez pacjenta” b) Jeśli tak to: umieść pacjenta na końcu kolejki 3. Zaplanuj następne zgłoszenie. 4. Obsłuż kolejne zdarzenie.
Pojawienie się dawcy	Zdarzenie jest generowane w momencie pojawienia się dawcy w punkcie krwiodawstwa. Czas między zgłoszeniem się kolejnych dawców jest zmienną losową o rozkładzie wykładniczym i średniej L.	1.Generuj czas pojawienia się kolejnego dawcy. 2. Dodaj jedną jednostkę krwi. 3. Zaplanuj czas przydatności pobranej krwi. 4. Zaplanuj kolejne zgłoszenie. 5. Obsłuż kolejne zdarzenie.

Dostawa krwi	Zdarzenie jest generowane w momencie dostawy krwi ze zlecenia lub awaryjnego zamówienia	1. Dodaj odpowiednią liczbę krwi z dostawy. 2. Zaplanuj czas przydatności dostarczonej krwi. 3. Obsłuż kolejne zdarzenie.
Utylizacja krwi	Zdarzenie występuję w momencie przeterminowania jednostek krwi w systemie	1. Usuń przeterminowane jednostki krwi. 2. Obsłuż kolejne zdarzenie.
Początek akcji promocyjnej	Zdarzenie jest generowane w momencie początku akcji promocyjnej	1. Zmniejsz czas pomiędzy pojawieniem się kolejnych dawców. 2. Zaplanuj zdarzenia końca akcji promocyjnej. 3. Zaplanuj zdarzenie kolejnej akcji promocyjnej. 4. Obsłuż kolejne zdarzenie.
Koniec akcji promocyjnej	Zdarzenie jest generowane po odpowiednim czasie od rozpoczęcia akcji promocyjnej	1. Przywróć normalny czas między pojawieniem się kolejnych dawców. 2. Obsłuż kolejne zdarzenie.

Zdarzenia warunkowe:

Zdarzenie	Opis	Algorytm
Wysłanie zlecenia na N nowych jednostek	W momencie gdy liczba dostępnych jednostek krwi spadnie do poziomu **R** lub niższego zostaje wysłanie zlecenie.	1. Sprawdź czy liczba dostępnym jednostek krwi większa lub równa R. a) Jeśli tak to kontynuuj. b) jeśli nie to wyślij zlecenie.
Pobranie krwi przez pacjenta	Pacjent pobiera odpowiednią ilość wymaganą do transfuzji.	1. Sprawdź czy dostępna odpowiednia liczba jednostek krwi: a) Jeśli tak to usuń odpowiednią liczbę jednostek krwi. b) Jeśli nie to wyślij awaryjne zamówienie.
Wysłanie awaryjnego zamówienia na Q jednostek	To zamówienie jest wysyłane w momencie gdy liczba potrzebnych jednostek krwi jest większa od aktualnego stanu zaopatrzenia.	1. Sprawdź czy liczba dostępnym jednostek krwi większa lub równa liczbie potrzebnych jednostek krwi. a) Jeśli tak to kontynuuj . b) jeśli nie to wyślij zamówienie.

5. Parametry wywołania programu

W momencie gdy użytkownik rozpocznie pracę ciągło następuje 10 symulacji w których: następuje wygenerowanie niezależnych wartości każdego z generatorów, a następne wykonanie symulacji określony liczbę razy (np. dla pojawienia się 20 tyś pacjentów).

6. Generatory

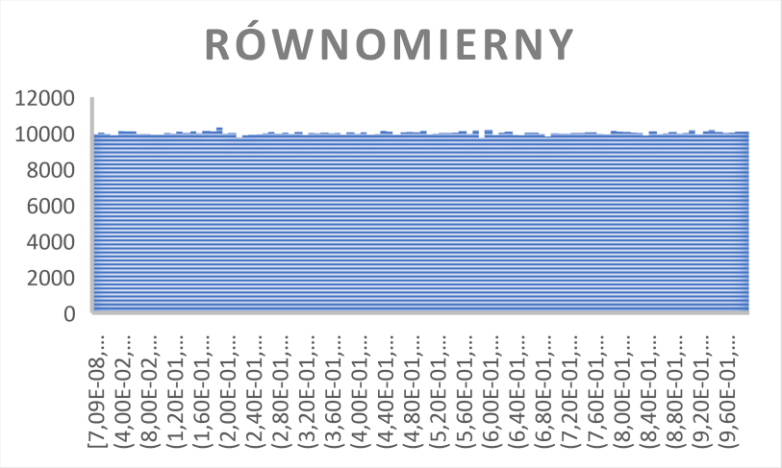
a) Opis zastosowanych generatorów liczb losowych z histogramami:

- Rozkład równomierny został wygenerowany za pomocą kodu:

```
const int two_31_ = 2147483648;
const int a_ = 16807;
const double q_ = 127773;
const int r_ = 2836;

UniformDistribution(){
double h = x_ / q_;
    x_ = a_*fmod(x_, q_) - h*r_;
    if (x_ < 0)
            x_ += (two_31_ - 1);
return x_ / (two_31_ - 1);
}
```

A jego histogram wygląda następująco:

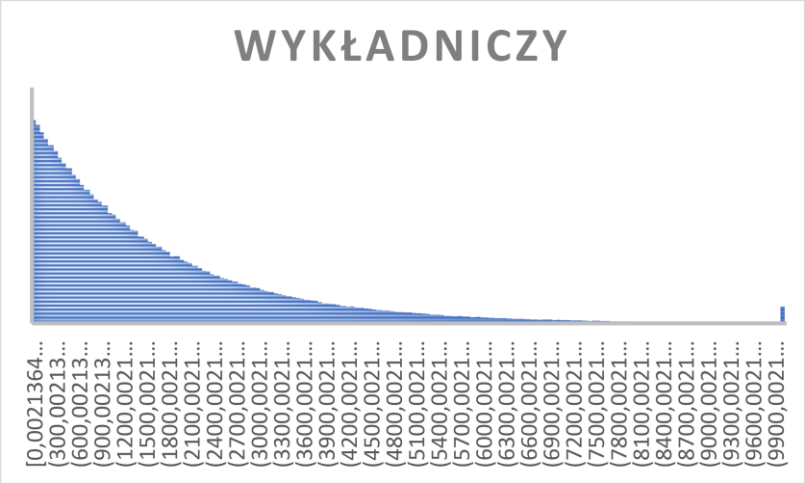


- Rozkład wykładniczy został wygenerowany za pomocą kodu:

```
ExponentialDistribution(const double lambda_) {
    return -pow(1 / lambda_, -1)*log(UniformDistribution());
```

}

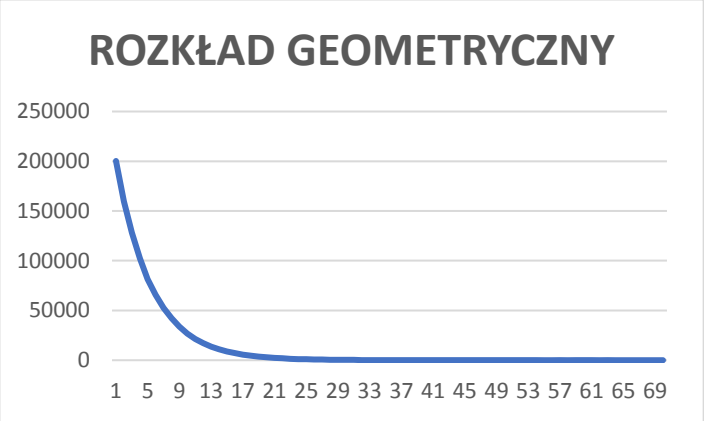
A jego histogram wygląda następująco:



- Rozkład geometryczny został wygenerowany za pomocą kodu:

```
const double w_ = 0.20;
GeometricDistribution() {
    int i = 0;
    do {
        i++;
    } while (UniformDistribution() < w_);
    return i;
}
```

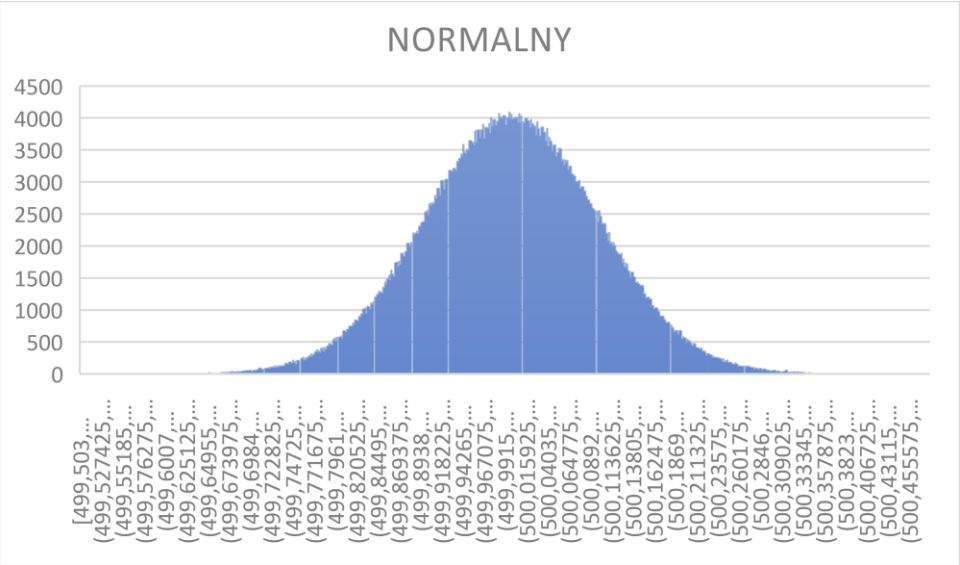
A jego histogram wygląda następująco:



- Rozkład normalny został wygenerowany za pomocą kodu:

```
NormalDistribution() {
    double first_draw;
    double second_draw;
    double x;
    do
    {
        first_draw = UniformDistribution();
        second_draw = UniformDistribution();
        x = -log(first_draw);
    } while (!(second_draw <= exp(-pow(x - 1, 2) / 2)));
    if (UniformDistribution() < 0.5)
    x = x * -1.0;
    return x * variance_ew_ + mean_e_;}
```

A jego histogram wygląda następująco:



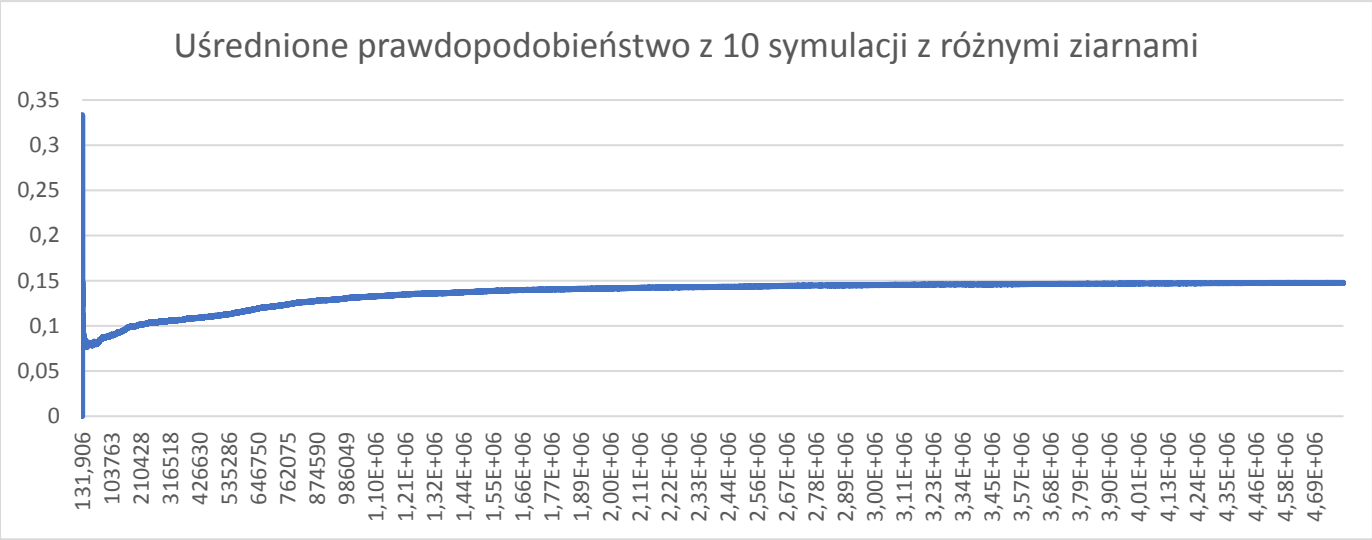
b) **Zapewnienie niezależności w różnych sekwencjach:**
Niezależność sekwencji w różnych symulacjach została zapewniona poprzez wygenerowanie różnych ziaren. Wygenerowałem 30 ziaren za pomocą funkcji:

```
for (int j = 0; j < 30; j++) {
    for (int i = 0; i <= 10000000; i++) {
           gen->UniformDistribution();
    }
    long long int Liczba2 = gen->CreateSeeds();
    std::cout << Liczba2 << std::endl;
    zapis << Liczba2 << std::endl;
}
```

Funkcja CreateSeeds() zwraca wylosowaną liczbę po odpowiedniej liczbie zadziałań generatora (u mnie 10 000 000). Ziarna są przechowywane w pliku „Seeds.txt”. Dla każdego zestawu parametrów przewidziałem po 10 symulacji z różnymi ziarnami.

7. **Krótki opis zastosowanej metody testowania i weryfikacji poprawności działania programu:**
   Zweryfikowałem działanie programu poprzez obserwowanie jego działania w pracy krokowej i poprzez porównanie z programami kolegów napisanymi w innych metodach. Również poprzez wykonanie 10 symulacji o różnych ziarnach i uśrednieniu parametrów. Z wykresu widać, że zbiega on do jednej stałej wartości.

8. **Wyniki symulacji:**
   a. **Wyznaczenie długości fazy początkowej:**
   Fazę początkową wyznaczam poprzez uśrednienie 10 symulacji i znalezienie punktu gdzie wykres zaczyna się stabilizować. W mojej symulacji następuje to dla czasu około 2 milionów.



   b. **Wyznaczenie parametrów:**
   Prawdopodobieństwo obliczane ze wzoru: $\frac{Próby\ wysłania\ awaryjnego\ zamówienia}{Liczba\ obsłużonych\ pacjentów}$
   Aby zwiększyć wiarygodność pomiaru liczba obsłużonych pacjentów jest wartością stała (symulacja wykonywana dla 20 tysięcy obsłużonych pacjentów)

R	N	T1	T2	Q	Przedział ufności min	$A_{uśrednione}$	Przedział ufności max
15	25	300	500	11	0.4415	0.4558	0.4700
45	25	300	500	11	0.3788	0.4252	0.4717
85	25	300	500	11	0.4300	0.4431	0.4562
45	50	300	500	11	0.4232	0.4309	0.4387
90	100	300	500	11	0.4212	0.4348	0.4484
Jak widać należy zmienić kolejne parametry, aby zmniejszyło się prawdopodobieństwo							
85	50	300	500	40	0.4152	0.4257	0.4361
85	50	300	500	80	0.4097	0.4222	0.4348
Ponieważ parametry nadal się nie poprawiają znaczy to, że trzeba zmienić czas przydatności krwi na większy, żeby się tak szybko nie utylizowała							
85	50	600	1000	40	0.3531	0.3663	0.3795
170	100	600	1000	40	0.3530	0.3645	0.3760
Dwukrotne zwiększenie pozostałych parametrów nie wpłynęło na zmianę prawdopodobieństwa, więc dalej będę zwiększał czas przydatności							
85	50	1200	2000	40	0.2891	0.3033	0.3175
170	100	1200	2000	40	0.2894	0.3017	0.3139
Kolejne dwukrotne zwiększenie czasu przydatności krwi							
85	50	2400	4000	40	0.2332	0.2491	0.2651
170	100	2400	4000	40	0.2407	0.2504	0.2600
Kolejne dwukrotne zwiększenie czasu przydatności krwi							
85	50	4800	8000	40	0.1882	0.1967	0.2051
170	100	4800	8000	40	0.1801	0.1958	0.2114
170	100	4800	8000	80	0.1861	0.1954	0.2047

Kolejne dwukrotne zwiększenie czasu przydatności krwi							
85	50	9600	16000	40	0.1314	0.1406	0.1499
170	100	9600	16000	40	0.1148	0.1254	0.1361
170	100	9600	16000	80	0.1146	0.1237	0.1328
Kolejne dwukrotne zwiększenie czasu przydatności krwi							
85	50	16200	32000	40	0.1052	0.1167	0.1282
170	100	16200	32000	40	0.0821	0.0873	0.0925
200	100	16200	32000	40	0.0772	0.0837	0.0901
250	100	16200	32000	40	0.0731	0.0794	0.0858
300	100	16200	32000	40	0.0773	0.0820	0.0867
Pomimo prób zmieniania pozostałych parametrów wynik nie ulega zmianie, więc musiałbym, żeby osiągnąć prawdopodobieństwo mniejsze niż 6% zwiększyć jeszcze trochę czas							
85	50	20000	35000	40	0.1073	0.1169	0.1266
170	100	20000	35000	40	0.0604	0.0711	0.0818
200	100	20000	35000	40	0.0662	0.0790	0.0917
250	100	20000	35000	40	0.0644	0.0705	0.0766
250	100	20000	35000	80	0.0684	0.0774	0.0863
350	150	20000	35000	40	0.0600	0.0684	0.0769

Dopiero dla takich o wiele większych wyników niż parametry zadane udało się osiągnąć prawdopodobieństwo około 6%. Dalsze zwiększanie czasu by skutkowało zapewne dalszym obniżeniem prawdopodobieństw.

c. Wyniki symulacji dla każdego przebiegu dla najlepszych parametrów (R=350, N=150, T1=20000, T2=35000, Q=40):

Numer przebiegu symulacji	Prawdopodobieństwo awaryjnego zgłoszenia	Prawdopodobieństwo utylizacji krwi
1	0.0737449	0.3769
2	0.06036	0.346015
3	0.0676142	0.384172
4	0.0724528	0.332846
5	0.0824102	0.368408
6	0.0617963	0.39671
7	0.0616325	0.389137
8	0.0568958	0.384886
9	0.0770562	0.378277
10	0.0702469	0.368461

$$Średnie\ prawdopodobieństwo\ awaryjnego\ zgłoszenia = \frac{Suma\ prawdopodobieństw}{10} = 0{,}0684$$

$$Średnie\ prawdopodobieństwo\ utylizacji\ krwi = \frac{Suma\ prawdopodobieństw}{10} = 0.3726$$

d. Wyniki końcowe:

parametr	Przedział ufności min	Średnia wartość	Przedział ufności max
Prawdopodobieństwo awaryjnego zgłoszenia	0.0600	0.0684	0.0769
Prawdopodobieństwo utylizacji krwi	0.3523	0.3726	0.3929

9. **Wnioski:**
   Aby otrzymać założone prawdopodobieństwo konieczna była znacząca zmiana parametrów zadanych, w przypadku czasu przydatności krwi była to zmiana 70-krotna co jest dużym odstępem od wartości z treści zadania.