

**LAPORAN PROYEK PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2023**

**GAME EPSILON DELTA SEBAGAI SARANA UNTUK MEMPERLAJARI  
KALKULUS (LIMIT) YANG MENARIK DAN MENYENANGKAN**



**Oleh Kelompok 32**

**Anggota:**

**Abdi Apriadi F1D022028**

**Lalu Romy Rahmad Amarta Putra F1D022058**

**Muhammad Khairul Amsal F1D022064**

**Ida Ayu Vinaya Anindya F1D022124**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2023**

**LEMBAR PENGESAHAN LAPORAN PROYEK**  
**PRAKTIKUM ALGORITMA DAN PEMROGRAMAN 2023**

1. Kelompok : 32
2. Judul Proyek : Game Epsilon Delta Sebagai Sarana untuk Mempelajari  
Kalkulus (Limit) yang Menarik dan Menyenangkan
3. Anggota Kelompok : Abdi Apriadi (F1D022028)  
Lalu Romy Rahmad Amarta Putra (F1D022058)  
Muhammad Khairul Amsal (F1D022064)  
Ida Ayu Vinaya Anindya (F1D022124)

Laporan proyek ini disusun sesuai dengan kaidah penyusunan yang telah ditentukan dan dibuat sebagai syarat mata kuliah Algoritma dan Pemrograman 2023.

Mataram, 21 Juni 2023

Telah diperiksa dan disahkan oleh:

**Koordinator Asisten**

**Asisten Pembimbing**

**Muhammad Dzulhi Raihan**  
**F1D021015**

**Muhammad Dzulhi Raihan**  
**F1D021015**

## 1.1. JUDUL

*Game Epsilon Delta* Sebagai Sarana untuk Mempelajari Kalkulus (Limit) yang Menarik dan Menyenangkan

## 1.2. LATAR BELAKANG

Matematika adalah bidang ilmu yang sangat penting dan berguna sehingga menjadi salah satu pelajaran wajib di sekolah maupun universitas. Matematika dapat membantu menyelesaikan permasalahan sehari-hari, pendidikan, dan sebagainya. Sebagai salah satu konsep dalam matematika, limit fungsi menjadi salah satu bahasan pokok dalam mata kuliah kalkulus. Konsep ini juga menjadi dasar dari konsep kalkulus lain seperti turunan dan integral. Salah satu cara dalam menyelesaikan permasalahan limit fungsi adalah dengan algoritma *epsilon-delta*.

Namun, mempelajari konsep seperti limit fungsi kurang diminati oleh pelajar karena permasalahan matematis dianggap sulit untuk diselesaikan. Di lain sisi, bermain *game* menjadi hal yang sangat digemari karena dinilai menarik dan menyenangkan. Jika hal tersebut terus dibiarkan, maka minat pelajar untuk mempelajari matematika akan semakin berkurang akibat kecanduan bermain *game*.

Dari permasalahan tersebut, kami mencoba memberikan solusi dengan membuat sebuah *game* bernama *Epsilon Delta* sebagai sarana untuk menarik minat pelajar dalam memahami konsep limit fungsi dengan algoritma *epsilon-delta*. Pelajar akan bermain *game* sambil belajar dalam waktu yang bersamaan. Dengan adanya *game* tersebut, diharapkan pelajar dapat memahami dan menguasai algoritma *epsilon-delta* sebagai salah satu cara menyelesaikan permasalahan mengenai limit fungsi dengan lebih menyenangkan.

## 1.3. DESKRIPSI PROGRAM

Program yang dikembangkan merupakan sebuah *game* yang dapat dimainkan oleh dua *player* untuk memecahkan sebuah permasalahan limit fungsi menggunakan algoritma *epsilon-delta*. Program dilengkapi dengan beberapa menu seperti tutorial, bermain (*play*), riwayat pertandingan (*match history*), mencari pertandingan yang sudah dimainkan (*find match*), dan menu keluar (*exit*). Jika *player* belum memahami algoritma *epsilon-delta* yang akan digunakan dan peraturan memainkan *game*, maka *player* dapat mempelajarinya terlebih dahulu pada menu tutorial.

Pada menu *play*, masing-masing *player* secara bergantian akan diminta untuk membuat dan menjawab soal satu sama lain. *Player* yang membuat soal atau *challenger* akan diminta memasukkan derajat *polynomial*, koefisien, konstanta, nilai limit yang mendekati  $x$ , menetapkan nilai *epsilon* dan *delta* serta *point* yang akan diperoleh oleh *player* lain jika

berhasil menjawab dengan benar. Kemudian *player* yang lain akan mencari *delta* dari persamaan *polynomial* yang diberikan.

Player juga dapat melihat *match history* atau riwayat pertandingan yang sudah dimainkan, di mana *point* yang didapatkan dari setiap *player* yang sudah bermain akan ditampilkan. Selain itu, player juga dapat mencari pertandingan yang telah dimainkan dengan memilih menu *find match*, di mana program akan meminta pengguna memasukkan nama player yang pertandingannya ingin dicari. Kemudian, pengguna dapat keluar dari program dengan memilih menu *exit*.

#### 1.4. ALGORITMA

1. Tampilkan halaman awal program berupa beberapa menu yaitu “PLAY”, “MATCH HISTORY”, “FIND MATCH”, dan “EXIT”
2. Jika *user* memilih opsi “PLAY!” maka fungsi untuk bermain akan dipanggil. Di dalam fungsi tersebut, *challenger* akan diminta untuk men-set soal beserta jawabannya. Kemudian, program akan meminta pengguna yang lain untuk menjawab soal yang diberikan *challenger*.
3. Program akan memanggil suatu fungsi yang akan mengecek jawaban dari *challenger* dan pihak yang menjawab. Jika pihak yang menjawab menjawab dengan benar maka ia akan mendapatkan poin sesuai dengan bobot soal. Sebaliknya, jika salah maka ia tidak mendapatkan poin. Jika *challenger* dan pihak yang menjawab memberikan jawaban yang benar maka *challenger* tidak mendapat poin. Selain itu, jika *challenger* memberikan jawaban yang salah maka poinnya akan dikurangi sesuai bobot soal.
4. Setelah satu permainan selesai, program akan menyimpan hasil permainan dalam suatu *linked-list*.
5. Program akan menanyakan apakah permainan akan dilanjutkan atau tidak. Jika iya program akan kembali ke proses nomor 2. Jika tidak, program akan kembali ke halaman awal.
6. Jika *user* memilih “MATCH HISTORY” maka fungsi menampilkan hasil permainan akan dipanggil. Selain itu jika *user* memilih “FIND MATCH” maka *user* akan diminta untuk memasukkan nama pemain yang akan dicari.
7. Jika *user* memilih “EXIT” maka program akan selesai.

## 1.5. PENJELASAN CODE

```
#include<iostream>
#include<cmath>
#include<iomanip>
#include<conio.h>
using namespace std;
```

*Script* di atas menandakan bahwa program mendeklarasikan beberapa *header file* seperti, “#include<iostream>” untuk menggunakan perintah “cin” dan “cout”, “#include<cmath>” untuk memungkinkan dilakukannya fungsi matematika dasar, “#include<iomanip>” untuk mengatur format input atau output seperti “setw” untuk mengatur lebar kolom, dan “#include<conio.h>” untuk menggunakan fungsi seperti “clrscr()” untuk membersihkan layar *cmd*. Program juga menggunakan *identifier std* untuk mempermudah penulisan perintah tanpa perlu menuliskan *prefix std* pada kode-kode program yang ada.

```
struct match_history{
    int score_p1;
    int score_p2;
    match_history* next;
};
```

*Script* di atas menandakan bahwa program mendeklarasikan sebuah variabel *struct* dengan nama “match\_history” yang memiliki beberapa member yaitu “score\_p1” dan “score\_p2” bertipe integer. Setiap objek “match\_history” akan merepresentasikan riwayat pertandingan *player* 1 dan *player* 2, serta pointer “\* next” akan mengakses entri data pertandingan berikutnya.

```
struct game{
    string p1;
    string p2;
    match_history* head;
};game matches[100];
void play(int& n_game);
void display_home(int& n_game);
```

*Script* di atas menandakan bahwa program mendeklarasikan sebuah variabel *struct* dengan nama “game” yang memiliki beberapa member yaitu “p1” dan “p2” bertipe *string*. Kemudian ada *pointer* “\* head” yang menunjuk ke objek “match\_history” yang merepresentasikan riwayat pertandingan “p1” dan “p2”, mendeklarasikan “array game matches[100]” yang dapat menyimpan sampai 100 pertandingan, dan mendeklarasikan fungsi “void play(int& n\_game)” dan “void display\_home(int& n\_game)”.

```
void insert_score(match_history*& head,int a,int b,int
indeks_game){
    match_history* match_baru=new match_history;
```

```

        match_baru->score_p1=a;
        match_baru->score_p2=b;
        match_baru->next=NULL;
        if(head==NULL){
            matches[indeks_game].head=match_baru;
        }
        else{
            match_history* current = head;
            while (current->next != NULL) {
                current = current->next;
            }
            current->next = match_baru;
        }
    }
}

```

Script di atas fungsi untuk menambahkan hasil pertandingan dari para *player* kedalam *single linked-list*. Fungsi tersebut bernama “insert\_score” dengan parameter “match\_history\*& head,int a,int b,int indeks\_game”. Setiap hasil pertandingan akan disimpan sesuai dengan indeks dari game yang dimainkan.

```

void menampilkan_soal(int degree, int arr[],double epsilon,int
limit){
    cout<<"Here is the Polynomial : \n"<<endl;
    cout<<"\t\t";
    if(degree==2){
        cout<<arr[0]<<"x^2 + ("<<arr[1]<<")x +
("<<arr[2]<<")"<<endl;
    }
    else{
        cout<<arr[0]<<"x + ("<<arr[1]<<")"<<endl;
    }
    cout<<"The Limit is Evaluated When X Approaches
"<<limit<<endl;
    cout<<"Your Task is to Find the Delta Given the Epsilon is
"<<epsilon<<endl;
}

```

Script di atas adalah fungsi untuk menampilkan soal dan meminta *player* yang bertugas menjawab untuk memberikan jawabannya. Fungsi tersebut bernama “menampilkan\_soal” dengan parameter “int degree, int arr[],double epsilon,int limit”. Program akan menampilkan soal berbentuk persamaan *polynomial* berdasarkan komponen yang telah dimasukkan *challenger* sebelumnya. Program juga menampilkan keterangan nilai limit yang mendekati x dan meminta *player* memasukkan nilai *delta* berdasarkan nilai *epsilon* yang ada.

```

bool mencocokkan (string& first, string& second) {
    if (first.size() < second.size()) {
        return false;
    }
    for (int i = 0; i <= first.size() - second.size(); ++i) {
        bool found = true;
        for (int j = 0; j < second.size(); ++j) {
            if (first[i + j] != second[j]) {
                found = false;
            }
        }
    }
}

```

```

        break;
    }
}
if (found) {
    return true;
}
return false;}

```

*Script* di atas adalah fungsi yang mencocokkan ukuran soal yang ditampilkan sesuai dengan ukuran derajat *polynomial* yang dimasukkan *challenger* sebelumnya. Fungsi tersebut bernama “mencocokkan” dengan parameter “string& first, string& second”. Kecocokan soal diukur dengan mengecek nilai dari variabel “found” yang dideklarasikan. Jika nilai variabel tersebut *true*, maka soal dikatakan cocok. Sedangkan jika nilai dari variabel “found=false”, maka soal tidak cocok.

```

void mencari_hasil_match(int n_game){
    system("cls");
    string name;
    cout<<" Find player name : ";
    cin>>name;
    bool found=1;
    for(int i=0;i<=n_game;i++){

        if(mencocokkan(matches[i].p1,name)||mencocokkan(matches[i].p2
, name)){

            found=0;
            cout<<"\n";
            cout<<"          |          MATCH RESULT
|"<<endl;
            cout<<"          |=====|"<<endl;
            cout<<"          |
"<<left<<setw(10)<<matches[i].p1<<"          |
"<<left<<setw(10)<<matches[i].p2<<"          |"<<endl;
            match_history* current=matches[i].head;
            while(true){
                cout<<"          |
"<<right<<setw(3)<<current->score_p1<<"          |
"<<right<<setw(3)<<current->score_p2<<"          |"<<endl;
                current=current->next;
                if(current==NULL)break;
            }
            cout<<"
|=====|"<<endl;
        }
    }
    if(found){
        cout<<"\t There is no such player name!"<<endl;
    }
    system("pause");
    system("cls");
    display_home(n_game);}

```

*Script* di atas adalah fungsi untuk menampilkan hasil pertandingan sesuai dengan nama *player* yang hasil pertandingannya ingin dicari. Fungsi tersebut bernama “mencari\_hasil\_match” dengan parameter “int n\_game”. Jika nama *player* yang

dimasukkan pengguna cocok dengan salah satu nama yang ada dalam daftar *player* yang telah bermain, maka program akan menampilkan hasil pertandingannya berupa skor lengkap dengan nama pemain (nama *player* tersebut dan *player* lawannya). Setelah itu, pengguna akan dialihkan kembali menuju tampilan awal dengan memanggil fungsi “display\_home(n\_game)”.

```
double find_answer(int degree,int arr[],double epsilon,int
evaluate){
    double hasil;
    if(degree==2){
        if(arr[1]*arr[1]-4*arr[0]*arr[2]==0){
            hasil=sqrt(epsilon);
        }
        else{
            double a=arr[0];
            double b=arr[1];
            double divide=b/a;
            double hasil_pengurangan = abs(divide+2*evaluate);
            hasil=epsilon/hasil_pengurangan;
        }
    }
    else{
        double b=abs(arr[0]);
        hasil=epsilon/b;
    }
    return hasil;}

```

*Script* di atas adalah fungsi yang di dalamnya dilakukan operasi untuk mencari nilai yang *delta* sesuai komponen soal yang dimasukkan *challenger*. Fungsi tersebut bernama “find\_answer” dengan parameter “int degree,int arr[],double epsilon,int evaluate”. Program akan melakukan operasi sesuai dengan rumus *epsilon-delta* yang hasilnya nanti akan menjadi tolak ukur kebenaran jawaban yang dimasukkan *player* yang bertugas menjawab soal.

```
bool check_answer(double delta,double jawaban){
    bool hasil;
    if(delta==jawaban){
        hasil =true;
    }
    else{
        hasil=false;
    }
    return hasil;}

```

*Script* di atas adalah fungsi untuk mengecek kebenaran jawaban yang dimasukkan *challenger* dan *player* yang bertugas menjawab. Fungsi tersebut bernama “check\_answer” berbentuk *Boolean* dengan parameter “double delta,double jawaban”. Di dalamnya dideklarasikan variabel bernama “hasil” bertipe *Boolean*. Nilai dari variabel “hasil” bergantung pada kesesuaian nilai variabel “delta” dengan variabel “jawaban”. Jika nilai variabel “delta” dengan variabel “jawaban” sesuai, maka



nilai variabel “hasil=true” yang berarti jawaban yang dimasukkan benar. Jika yang terjadi adalah sebaliknya, maka nilai variabel “hasil=false” sehingga jawaban yang dimasukkan salah.

```
void match_result(int n_game,int score_p1_part1,int
score_p2_part1,int score_p1,int score_p2){
    cout <<left << setw(45)<<matches[n_game].p1+" : The
Challenger "<<"+"<<score_p1_part1<<" "<<endl;
    cout <<left << setw(45)<<matches[n_game].p2+" :
"<<"+"<<score_p2_part1<<endl;
    cout <<left << setw(45)<<matches[n_game].p1+" :
"<<"+"<<score_p1<<endl;
    cout <<left << setw(45)<<matches[n_game].p2+" : The
Challenger "<<"+"<<score_p2<<" "<<endl;}
```

Script di atas adalah fungsi untuk menampilkan hasil pertandingan yang telah dilakukan para *player*. Fungsi tersebut bernama “match\_result” dengan parameter “int n\_game,int score\_p1\_part1,int score\_p2\_part1,int score\_p1,int score\_p2”. Fungsi ini akan menampilkan skor yang diperoleh oleh *player* 1 dan 2 beserta nama dari masing-masing *player*.

```
void menampilkan_match_history(int n_game){
    system("cls");
    if(n_game==0){
        cout<<"\t There is no game played yet!"<<endl;}
    else{
        cout<<"\n";
        cout<<"          |                      MATCH RESULT
|"<<endl;
        cout<<"          |=====|"<<endl;
        for(int i=0;i<n_game;i++){
            cout<<"          |
"<<left<<setw(10)<<matches[i].p1<<"          |
"<<left<<setw(10)<<matches[i].p2<<"          |"<<endl;
            match_history* current=matches[i].head;
            while(true){
                cout<<"          |
"<<right<<setw(3)<<current->score_p1<<"          |
"<<right<<setw(3)<<current->score_p2<<"          |"<<endl;
                current=current->next;
                if(current==NULL)break;
            }
            cout<<"
|=====|"<<endl;
        }
    }
    system("pause");
    system("cls");
    display_home(n_game);}
```

Script di atas adalah fungsi untuk menampilkan riwayat pertandingan yang telah dimainkan. Fungsi tersebut bernama “menampilkan\_match\_history” dengan parameter “int n\_game”. Jika belum ada pertandingan yang dimainkan maka program akan menampilkan kalimat “There is no game played yet!”. Jika sudah ada yang

pertandingan yang dimainkan, maka program akan menampilkan hasil dari pertandingan tersebut berupa skor dan nama dari para *player*. Setelah itu, pengguna akan dialihkan kembali ke menu awal dengan memanggil fungsi “display\_home(n\_game”.

```
void tutorial(int n_game){
    system("cls");
    cout<<"\t\t\t>>TUTORIAL<<"<<endl;
    cout<<"|=====|
=====| "<<endl;
    cout<<"|\t\t>>FORMAL DEFINITION OF LIMIT<<
| "<<endl;
    cout<<"| Let f(x) be a function defined on an open interval
around x0. | "<<endl;
    cout<<"| We say that the limit of f(x) as x approaches x0 is
L, i.e | "<<endl;
    cout<<"|
| ";
    cout<<"\n|\t xâ†'x0 lim f(x) = L,
| "<<endl;
    cout<<"|
| ";
    cout<<"\n| if for every Îµ>0 (epsilon>0),
| "<<endl;
    cout<<"| There exists Î´>0 (delta>0) such that for all x
where | "<<endl;
    cout<<"|
| ";
    cout<<"\n|\t0 < | x - x0 | < Î´ ==> | f(x) - L | < Îµ.
| "<<endl;
    cout<<"|
| ";
    cout<<"\n| For more detailed explanation about this
defintion, | "<<endl;
    cout<<"| You can check on this link
https://youtu.be/kfF40MiS7zA | "<<endl;
    cout<<"|=====|
=====| "<<endl;
    cout<<"\n The main goal of this game is to introduce you to
calculus, "<<endl;
    cout<<" Especially to the formal definition of limit"<<endl;
    cout<<" This is a dual-player game"<<endl;
    cout<<" In each match, both player will play as a challenger
alternately "<<endl;
    cout<<" The challenger will set up the function including the
epsilon"<<endl;
    cout<<" In turn, the opponent will guess the delta given the
epsilon with respect to the function"<<endl;
    cout<<" REMEMBER, This game only provide up to 2 degree of
polynomial function "<<endl;
    cout<<" Therefore, there are only two function which is
linear function and quadratic function"<<endl;
    cout<<" You also need to know that if the challenger insert
the incorrect delta given the epsilon,"<<endl;
    cout<<" There will be a punishment in term of score,"<<endl;
    cout<<" The weight of punishment will be according to the
weight of the question"<<endl;
    cout<<" If the opponent guessed incorrectly and if the error
is in between 1*10^-6,"<<endl;
```

```

        cout<<" Then the opponent will gain half of the question
weight"<<endl;
        cout<<"\n Alright, guess you're good to go!"<<endl;
        system("pause");
        system("cls");
        display_home(n_game);}

```

*Script* di atas adalah fungsi tutorial yang akan memberikan petunjuk kepada pengguna mengenai dasar dari algoritma *epsilon-delta*, deskripsi *game*, dan peraturan dalam bermain. Pada fungsi ini ditampilkan rumus *epsilon-delta*, deskripsi *game* seperti “The main goal of this game is to introduce you to calculus, Especially to the formal definition of limit, This is a dual-player game, In each match, both player will play as a challenger alternately, The challenger will set up the function including the epsilon, and In turn, the opponent will guess the delta given the epsilon with refers to the function”, dan peraturan dalam *game* seperti “REMEMBER, This game only provide up to 2 degree of polynomial function, Therefore, there are only two function which is linear function and quadratic function, You also need to know that if the challenger insert the incorrect delta given the epsilon, There will be a punishment in term of score, The weight of punishment will be according to the weight of the question, If the opponent guessed incorrectly and if the error is in between  $1 \cdot 10^{-6}$ , Then the opponent will gain half of the question weight”.

Setelah selesai membaca tutorial, ketika pengguna menekan tombol *enter*, layar pada *cmd* akan dibersihkan. Kemudian, program akan memanggil fungsi “display\_home(n\_game)” dan menjalankannya. Jika sudah kembali ke tampilan awal, pengguna dapat memilih opsi *play* dan mulai memainkan *game*.

```

void display_home(int& n_game){
    int pilih=1;
    int a;
    while(true){
        if(pilih==1){
            system("cls");
            cout<<" Welcome to The Epsilon-Delta Game!\n";
            cout<<"|=====|"<<endl;
            cout<<"| ==>>[      PLAY!      ]<=====|"<<endl;
            cout<<"|      [ MATCH HISTORY ]      |"<<endl;
            cout<<"|      [   FIND MATCH   ]      |"<<endl;
            cout<<"|      [   TUTORIAL    ]      |"<<endl;
            cout<<"|      [      EXIT      ]      |"<<endl;
            cout<<"|=====|"<<endl;
        }else if(pilih==2){
            system("cls");
            cout<<" Welcome to The Epsilon-Delta Game!\n";
            cout<<"|=====|"<<endl;
            cout<<"|      [      PLAY!      ]      |"<<endl;
            cout<<"| ==>>[ MATCH HISTORY ]<=====|"<<endl;
            cout<<"|      [   FIND MATCH   ]      |"<<endl;

```

```

        cout<<"|          [    TUTORIAL    ]          |"<<endl;
        cout<<"|          [      EXIT      ]          |"<<endl;
        cout<<"|=====|"<<endl;
    }else if(pilih==3){
        system("cls");
        cout<<" Welcome to The Epsilon-Delta Game!\n";
        cout<<"|=====|"<<endl;
        cout<<"|          [      PLAY!      ]          |"<<endl;
        cout<<"|          [ MATCH HISTORY ]          |"<<endl;
        cout<<"| ==>>[    FIND MATCH    ]<<==>> |"<<endl;
        cout<<"|          [    TUTORIAL    ]          |"<<endl;
        cout<<"|          [      EXIT      ]          |"<<endl;
        cout<<"|=====|"<<endl;
    }else if(pilih==4){
        system("cls");
        cout<<" Welcome to The Epsilon-Delta Game!\n";
        cout<<"|=====|"<<endl;
        cout<<"|          [      PLAY!      ]          |"<<endl;
        cout<<"|          [ MATCH HISTORY ]          |"<<endl;
        cout<<"|          [    FIND MATCH    ]          |"<<endl;
        cout<<"| ==>>[    TUTORIAL    ]<<==>> |"<<endl;
        cout<<"|          [      EXIT      ]          |"<<endl;
        cout<<"|=====|"<<endl;
    }else if(pilih==5){
        system("cls");
        cout<<" Welcome to The Epsilon-Delta Game!\n";
        cout<<"|=====|"<<endl;
        cout<<"|          [      PLAY!      ]          |"<<endl;
        cout<<"|          [ MATCH HISTORY ]          |"<<endl;
        cout<<"|          [    FIND MATCH    ]          |"<<endl;
        cout<<"|          [    TUTORIAL    ]          |"<<endl;
        cout<<"| ==>>[      EXIT      ]<<==>> |"<<endl;
        cout<<"|=====|"<<endl;
    }
    a=getch();
    if(a==72){
        pilih--;
        if(pilih<1){
            pilih=5;
        }
    }else if(a==80){
        pilih++;
        if(pilih>5){
            pilih=1;
        }
    }else if(a==13){
        break;
    }
}
if(pilih==1){
    play(n_game);
}
else if(pilih==2){
    menampilkan_match_history(n_game);
}
else if(pilih==3){
    mencari_hasil_match(n_game);
}
else if(pilih==4){
    tutorial(n_game);
}
}

```

```
else {  
    return;}}
```

*Script* di atas adalah fungsi yang akan menampilkan bagian awal program. Fungsi tersebut bernama “display\_home” dengan parameter “&n\_game” dalam bentuk integer. Di dalam fungsi awalnya dideklarasikan sebuah variabel bernama “pilih” dengan nilai awal 1 dan variabel “a” yang berfungsi sebagai pengganti spasi. Program menggunakan *statement control switch-case* untuk mengecek perintah yang diberikan oleh pengguna ketika memilih beberapa opsi yang telah disediakan.

Pengguna diberikan beberapa opsi ketika berada di awal program, di mana terdapat opsi *play* yang akan terpilih ketika nilai variabel “pilih=1”, *match history* yang akan terpilih ketika nilai variabel “pilih=2”, *find match* yang akan terpilih ketika nilai variabel “pilih=3”, tutorial yang akan terpilih ketika nilai variabel “pilih=4” dan *exit* yang akan terpilih ketika nilai variabel “pilih=5”. Setelah pengguna memilih salah satu opsi, program akan memanggil fungsi sesuai pilihan pengguna. Jika pengguna memilih opsi *play*, program akan memanggil fungsi “play(n\_game)” dan menjalankan fungsi tersebut. Jika pengguna memilih opsi *match history*, program akan memanggil fungsi “menampilkan\_match\_history(n\_game)” dan menjalankan fungsi tersebut. Jika pengguna memilih opsi *match history*, program akan memanggil fungsi “menampilkan\_match\_history(n\_game)” dan menjalankan fungsi tersebut. Jika pengguna memilih opsi *find match*, program akan memanggil fungsi “mencari\_hasil\_match(n\_game)” dan menjalankan fungsi tersebut. Jika pengguna memilih opsi tutorial, program akan memanggil fungsi “tutorial(n\_game)” dan menjalankan fungsi tersebut. Dan jika pengguna memilih opsi *exit* maka program akan ditutup.

```
void play(int& n_game){  
    system("cls");  
    cout<<"==>Please Insert the Challenger 1 Name : ";  
    cin>>matches[n_game].p1;  
    cout<<"==>Please Insert the Challenger 2 Name : ";  
    cin>>matches[n_game].p2;  
    system("cls");  
    while(true){  
        int degree,evaluate,weight,score_p1,score_p2;  
        int score_p1_part1,score_p2_part1;  
        double delta,epsilon,answer,real_answer;  
        char play_again;  
        while(true){  
            cout<<"Alright, Remember Not to Show the Screen  
While You're Setting Up the Challenge! \n";  
            cout<<"Hey "<<matches[n_game].p1<<"! Please  
Insert the Degree of Your Polynomial [1 or 2] : ";
```

```

        cin>>degree;
        if (degree==1||degree==2)break;
    }
    int arr[degree+1];
    for(int i=0;i<degree+1;i++){
        if (degree==1){
            if (i==0){
                cout<<"Please Insert the Coefficient : ";
                cin>>arr[i];
            }
            else{
                cout<<"Please Insert the Constant : ";
                cin>>arr[i];
            }
        }
        else{
            if (i==2){
                cout<<"Please Insert the Constant : ";
                cin>>arr[i];
            }
            else{
                cout<<"Please Insert the Coefficient : ";
                cin>>arr[i];
            }
        }
    }
    cout<<"The Limit is Evaluated When X Approaches : ";
    cin>>evaluate;
    cout<<"Set the Value of Your Epsilon : ";
    cin>>epsilon;
    cout<<"According to Your Epsilon,Insert your Delta : ";
    cin>>delta;
    while(true){
        cout<<"How Many Points You Want to Give this Problem? The Weight Must be in Interval[1,20] : ";
        cin>>weight;
        if (weight>0&&weight<21)break;
    }
    system("cls");
    cout<<"Now, Your Turn "<<matches[n_game].p2<<"!, Insert Your Delta According to the Given Polynomial ";
    menampilkan_soal(degree,arr,epsilon,evaluate);
    cin>>answer;
    system("cls");
    cout<<"Here is the result : \n";

    if (check_answer(answer,find_answer(degree,arr,epsilon,evaluate))){
        cout <<left << setw(45)<<matches[n_game].p2 + " Got the Correct Answer!" <<"+"<<weight<<endl;
        score_p2_part1=weight;
    }
    else
        if (check_answer(answer,find_answer(degree,arr,epsilon,evaluate))==false&& abs(answer-find_answer(degree,arr,epsilon,evaluate))<=1e-06){

```

```

        cout <<left << setw(45)<<matches[n_game].p2+"
Nearly Got the Correct Answer!"<<"+ "<<weight/2<<endl;
        score_p2_part1=weight/2;
    }
    else{
        cout <<left << setw(45)<<matches[n_game].p2+" Got
the Wrong Answer!"<<"+0"<<endl;
        score_p2_part1=0;
    }

    if(check_answer(delta,find_answer(degree,arr,epsilon,evaluate
))&&abs(answer-find_answer(degree,arr,epsilon,evaluate))>1e-06){
        cout <<left << setw(45)<<matches[n_game].p1+"
Insert the Correct Answer"<<"+ "<<weight<<endl;
        score_p1_part1=weight;
    }
    else
    if(check_answer(delta,find_answer(degree,arr,epsilon,evaluate))&&ab
s(answer-find_answer(degree,arr,epsilon,evaluate))<=1e-06){
        cout <<left << setw(45)<<matches[n_game].p1+"
Insert the Correct Answer"<<"+0"<<endl;
        score_p1_part1=0;
    }
    else{
        cout <<left << setw(45)<<matches[n_game].p1+"
Insert the Wrong Answer!"<<"+ "- "<<weight<<endl;
        score_p1_part1=-weight;
    }

    insert_score(matches[n_game].head,score_p1_part1,score_p2_par
t1,n_game);
    system("pause");
    system("cls");
    cout<<"Now, It's Your Turn, " <<matches[n_game].p2<< ",
Please Set Up Your Challenge! ";
    while(true){
        cout<<"Alright, Remember Not to Show the Screen
While You're Setting Up the Challenge! \n";
        cout<<"Hey " <<matches[n_game].p2<<"! Please
Insert the Degree of Your Polynomial [1 or 2] : ";
        cin>>degree;
        if(degree==1||degree==2)break;
    }
    for(int i=0;i<degree+1;i++){
        if(degree==1){
            if(i==0){
                cout<<"Please Insert the Coefficient :
";
                cin>>arr[i];
            }
            else{
                cout<<"Please Insert the Constant :
";
                cin>>arr[i];
            }
        }
        else{
            if(i==2){
                cout<<"Please Insert the Constant :
";
                cin>>arr[i];

```

```

        }
        else{
            cout<<"Please Insert the Coefficient
: ";
            cin>>arr[i];
        }
    }
    cout<<"The Limit is Evaluated When X Approaches : ";
    cin>>evaluate;
    cout<<"Set the Value of Your Epsilon : ";
    cin>>epsilon;
    cout<<"According to Your Epsilon,Insert your Delta : ";
    cin>>delta;
    while(true){
        cout<<"How Many Points You Want to Give this
Problem? The Weight Must be in Interval[1,20] : ";
        cin>>weight;
        if(weight>0&&weight<21)break;
    }
    system("cls");
    cout<<"Now, Your Turn "<<matches[n_game].p2<<"!, Insert
Your Delta According to the Given Polynomial ";
    menampilkan_soal(degree,arr,epsilon,evaluate);
    cin>>answer;
    system("cls");
    cout<<"Here is the result : \n";

    if(check_answer(answer,find_answer(degree,arr,epsilon,evaluate))){
        cout <<left << setw(45)<<matches[n_game].p1+" Got
the Correct Answer!"<<"+"<<weight<<endl;
        score_p1=weight;
    }
    else
    if(check_answer(answer,find_answer(degree,arr,epsilon,evaluate))==false&&abs(answer-find_answer(degree,arr,epsilon,evaluate))<=1e-06){
        cout <<left <<
setw(45)<<matches[n_game].p1+"Nearly Got the Correct
Answer!"<<"+"<<weight/2<<endl;
        score_p1=weight/2;
    }
    else{
        cout <<left << setw(45)<<matches[n_game].p1+" Got
the Wrong Answer!"<<"+"<<0<<endl;
        score_p1=0;
    }

    if(check_answer(delta,find_answer(degree,arr,epsilon,evaluate))&&abs(answer-find_answer(degree,arr,epsilon,evaluate))>1e-06){
        cout <<left << setw(45)<<matches[n_game].p2+"
Insert the Correct Answer!"<<"+"<<weight<<endl;
        score_p2=weight;
    }
    else
    if(check_answer(delta,find_answer(degree,arr,epsilon,evaluate))&&abs(answer-find_answer(degree,arr,epsilon,evaluate))<=1e-06){
        cout <<left << setw(45)<<matches[n_game].p2+"
Insert the Correct Answer!"<<"+"<<0<<endl;
        score_p2=0;
    }
}

```



```

        else{
            cout <<left << setw(45)<<matches[n_game].p2+"
Insert the Wrong Answer!"<<"-"<<weight<<endl;
            score_p2=-weight;
        }

        insert_score(matches[n_game].head,score_p1,score_p2,n_game);
        system("pause");
        system("cls");
        cout<<"\t\tThe Challenge Has Ended!";
        cout<<"\nHere is the Challenge Result!"<<endl;

        match_result(n_game,score_p1_part1,score_p2_part1,score_p1,score_p2);
        cout<<"\nYou Want to Play Again?";
        cin>>play_again;
        system("pause");
        system("cls");
        if(play_again!='y'){
            n_game++;
            display_home(n_game);
        }
    }
}

```

Script di atas adalah fungsi untuk menampilkan proses memainkan game *epsilon-delta*. Pada saat awal memasuki fungsi ini, program akan meminta pengguna memasukkan nama dari *player* 1 dan 2. Kemudian, *player* 1 akan bertindak sebagai *challenger* dan program akan memintanya membuat soal dengan memasukkan komponen soal persamaan *polynomial* mulai dari derajat *polynomial*, koefisien, konstanta, nilai limit yang mendekati  $x$ , nilai *epsilon*, nilai *delta* berdasarkan *epsilon*, dan bobot skor yang dapat diperoleh *player* lain berhasil menjawab dengan benar.

Selanjutnya, program akan menampilkan soal persamaan *polynomial* yang terbentuk berdasarkan komponen yang telah dimasukkan. Program akan meminta *player* 2 untuk menjawab pertanyaan dengan memasukkan nilai *delta* sesuai persamaan *polynomial* dan *epsilon* yang ada. Kemudian, program akan menampilkan hasil apakah jawaban yang dimasukkan benar atau salah.

Jika *player* 2 memasukkan jawaban yang tepat, maka akan memperoleh nilai sesuai dengan bobot yang telah ditetapkan *challenger*. Namun, jika salah maka tidak akan mendapatkan penambahan skor. Berlaku bagi *challenger* atau pembuat soal jika salah memasukkan nilai *delta* ketika membuat soal, maka akan diberikan pengurangan nilai sesuai bobot yang telah ditetapkan. Jika *challenger* memasukan nilai *delta* yang benar tidak mendapat nilai tambahan.

Setelah menampilkan skor, permainan akan beralih dan *player* 2 akan bertindak sebagai *challenger*. Program akan memintanya membuat soal dengan memasukkan

komponen soal persamaan *polynomial* mulai dari derajat *polynomial*, koefisien, konstanta, nilai limit yang mendekati x, nilai *epsilon*, nilai *delta* berdasarkan *epsilon*, dan bobot skor yang dapat diperoleh *player* lain berhasil menjawab dengan benar.

Selanjutnya, program akan menampilkan soal persamaan *polynomial* yang terbentuk berdasarkan komponen yang telah dimasukkan. Program akan meminta *player* 1 untuk menjawab pertanyaan dengan memasukkan nilai *delta* sesuai persamaan *polynomial* dan *epsilon* yang ada. Kemudian, program akan menampilkan hasil apakah jawaban yang dimasukkan benar atau salah.

Jika *player* 1 memasukkan jawaban yang tepat, maka akan memperoleh nilai sesuai dengan bobot yang telah ditetapkan *challenger*. Namun, jika salah maka tidak akan mendapatkan penambahan skor. Berlaku bagi *challenger* atau pembuat soal jika salah memasukkan nilai delta ketika membuat soal, maka akan diberikan pengurangan nilai sesuai bobot yang telah ditetapkan. Jika *challenger* memasukan nilai *delta* yang benar tidak mendapat nilai tambahan.

Setelah kedua *player* sudah bergantian sebagai *challenger* dan menjawab soal. Program akan menanyakan apakah para *player* ingin bermain kembali. Jika iya, maka akan dialihkan ke bagian membuat soal dan kembali bergantian menjadi *challenger* dan penjawab soal. Jika tidak, maka *player* akan dialihkan kembali menuju menu awal dengan memanggil fungsi “`display_home(n_game)`”.

```
int main(){
int n_game=0;
display_home(n_game);

return 0;}
```

*Script* di atas menandakan bahwa program memiliki fungsi “main” dengan beberapa perintah di dalamnya seperti mendeklarasikan variabel “n\_game” dalam bentuk integer dengan nilai awal 0. Memanggil fungsi “`display_home(n_game)`” untuk dijalankan dan perintah “`return 0`” sebagai tanda bahwa eksekusi program telah berakhir.

## 1.6. OUTPUT PROGRAM

```
Welcome to The Epsilon-Delta Game!
=====
===>>[    PLAY!    ]<<===
      [ MATCH HISTORY ]
      [  FIND MATCH  ]
      [   TUTORIAL   ]
      [    EXIT      ]
=====
```

**Gambar 1.1** Tampilan Awal *Game Epsilon Delta*

Pada **Gambar 1.1** merupakan tampilan awal dari program *game Epsilon Delta*. Pada awal program akan ditampilkan beberapa menu seperti *play*, *match history*, *find match*, *tutorial*, dan *exit*. Pengguna dapat memilih salah satu dari menu yang ada dan program akan mengalihkan pengguna menuju tampilan menu sesuai dengan pilihannya. Jika pengguna memilih menu *exit*, maka program akan selesai.

```
>>TUTORIAL<<
=====
>>FORMAL DEFINITION OF LIMIT<<
Let f(x) be a function defined on an open interval around x0.
We say that the limit of f(x) as x approaches x0 is L, i.e


$$\forall \epsilon > 0 \lim_{x \rightarrow x_0} f(x) = L,$$


if for every  $\epsilon > 0$  (epsilon>0),
There exists  $\delta > 0$  (delta>0) such that for all x where


$$0 < |x - x_0| < \delta \implies |f(x) - L| < \epsilon.$$


For more detailed explanation about this definition,
You can check on this link https://youtu.be/kff40MiS7zA
=====

The main goal of this game is to introduce you to calculus,
Especially to the formal definition of limit
This is a dual-player game
In each match, both player will play as a challenger alternately
The challenger will set up the function including the epsilon
In turn, the opponent will guess the delta given the epsilon with respect to the function
REMEMBER, This game only provide up to 2 degree of polynomial function
Therefore, there are only two function which is linear function and quadratic function
You also need to know that if the challenger insert the incorrect delta given the epsilon,
There will be a punishment in term of score,
The weight of punishment will be according to the weight of the question
If the opponent guessed incorrectly and if the error is in between  $1 \cdot 10^{-6}$ ,
Then the opponent will gain half of the question weight

Alright, guess you're good to go!
Press any key to continue . . .
```

**Gambar 1.2** Tampilan Menu Tutorial

Pada **Gambar 1.2** merupakan tampilan dari menu tutorial *game Epsilon Delta*. Pengguna yang belum memahami algoritma *epsilon-delta* dalam teorema limit fungsi dalam mempelejarinya terlebih dahulu pada menu ini. Menu tutorial memberikan penjelasan mengenai algoritma *epsilon-delta* dan aturan dalam memainkan game.

```
==>Please Insert the Challenger 1 Name : abdi
==>Please Insert the Challenger 2 Name : apriadi
```

**Gambar 1.3** Tampilan Menu *Play* (Memasukkan Nama *Player*)

Pada **Gambar 1.3** merupakan tampilan awal menu *play*. Di bagian ini, program akan meminta pengguna memasukkan nama dari *player* yang akan bermain. *Player* yang bermain disebut dengan *challenger*, di mana terdapat *challenger 1* dan *challenger 2*.

```
Alright, Remember Not to Show the Screen While You're Setting Up the Challenge!
Hey abdi! Please Insert the Degree of Your Polynomial [1 or 2] : 1
Please Insert the Coefficient : 1
Please Insert the Constant : 2
The Limit is Evaluated When X Approaches : 3
Set the Value of Your Epsilon : 4
According to Your Epsilon, Insert your Delta : 5
How Many Points You Want to Give this Problem? The Weight Must be in Interval[1,20] : 20
```

**Gambar 1.4** Tampilan Menu *Play* (Pembuatan Soal)

Pada **Gambar 1.4** merupakan tampilan dari menu *play* saat *challenger* membuat soal. *Challenger* akan bergantian membuat soal untuk dijawab oleh *player* lain secara bergantian masing-masing 1 kali dalam satu ronde. Pembuat soal akan diminta untuk memasukkan komponen *polynomial*, seperti derajat *polynomial*, koefisien, konstanta, nilai  $x$  yang mendekati limit, menetapkan *epsilon* dan *delta*, serta bobot *point* yang akan diperoleh *challenger* lain jika berhasil menjawab dengan benar.

```
Now, Your Turn apriadi!, Insert Your Delta According to the Given Polynomial Here is the Polynomial :
1x + (2)
The Limit is Evaluated When X Approaches 3
Your Task is to Find the Delta Given the Epsilon is 4
4
```

**Gambar 1.5** Tampilan Tampilan Menu *Play* (Menjawab Soal)

Pada **Gambar 1.5** merupakan tampilan dari menu *play* saat *challenger* menjawab soal yang telah dibuat oleh *challenger* lain. Program akan menampilkan soal kemudian meminta *challenger* untuk memasukkan nilai *delta* berdasarkan nilai *epsilon* yang dimasukkan pembuat soal.

```
Here is the result :
apriadi Got the Correct Answer!           +20
abdi Insert the Wrong Answer!             -20
Press any key to continue . . .
```

**Gambar 1.6** Tampilan Menu *Play* (Hasil Penilaian)

Pada **Gambar 1.6** merupakan tampilan dari menu *play* saat penampilan skor yang didapat ketika *challenger* menjawab soal dari *challenger* lain. Program akan menampilkan skor untuk masing-masing *challenger*, di mana *challenger* yang berperan

menjawab dan jawabannya benar akan mendapatkan skor sesuai dengan bobot nilai yang diberikan pembuat soal dan jika salah akan mendapat skor 0. Sedangkan, pembuat soal tidak mendapat skor tambahan sesuai dengan bobot nilai yang ditentukan jika jawaban yang dimasukkan ketika membuat soal benar, namun akan mendapat pengurangan skor sesuai dengan bobot nilai yang ditentukan jika kunci jawaban yang dimasukkan pada saat membuat soal salah.

MATCH RESULT	
abdi	apriadi
-20	20
6	-6

Press any key to continue . . .

**Gambar 1.7** Tampilan Menu *Match History*

Pada **Gambar 1.7** merupakan tampilan dari menu *match history*. Pada bagian ini, program akan menampilkan hasil pertandingan yang telah dimainkan berupa nama kedua *player* beserta skor yang diperoleh keduanya ketika bermain.

Find player name : abdi

MATCH RESULT	
abdi	apriadi
-20	20
6	-6

Press any key to continue . . .

**Gambar 1.8** Tampilan Menu *Find Match*

Pada **Gambar 1.8** merupakan tampilan dari menu *find match*. Pada bagian ini, program akan menampilkan hasil pertandingan yang ingin dicari oleh pengguna dengan memasukkan nama dari *player* yang hasil pertandingannya ingin dicari.

## 1.7. KESIMPULAN

Berdasarkan hasil pembahasan di atas maka dapat diambil kesimpulan sebagai berikut:

1. Pembuatan program seperti *game* dapat dibuat dengan kaidah-kaidah penyusunan program seperti penggunaan program dasar, *statement control*, *looping*, *function*, *struct*, *pointer* dan *linked-list* dalam bahasa pemrograman C++.
2. Konsep matematika seperti limit fungsi dengan menerapkan algoritma *epsilon-delta* dapat menjadi lebih menarik dan menyenangkan untuk dipelajari ketika diimplementasikan dalam sebuah permainan atau *game*.
3. Kecanduan bermain game akan menjadi positif jika game yang dimainkan dapat memberikan tambahan pengetahuan bagi pemainnya. Bermain game sambil belajar adalah kegiatan yang tepat untuk mewujudkannya.

## 1.8. REFERENSI

Toheri. (2016). Kalkulus Turunan. Cirebon: eduvision.

Handayani, N.F. & Mahrita. (2021). FAKTOR PENYEBAB KESULITAN BELAJAR MATEMATIKA PADA SISWA KELAS IV DI SDN JAWA 2 MARTAPURA KABUPATEN BANJAR. Banjarmasin: Program Studi Pendidikan Guru Sekolah Dasar; Bimbingan Konseling Universitas Achmad Yani Banjarmasin.

Khasanah, F. (2019). Pembelajaran Limit Fungsi Berbasis Android Aplikasi *Math Expert*. Malang: Program Studi Pendidikan Matematika Universitas Wisnuwardhana Malang.

Oktaviyanthi, R. & Agus, R.V. (2021). Guided Worksheet Formal Definition of Limit: An Instrument Development Process. Banten: Universitas Serang Raya.

Bahar, E.E. (2014). ANALISIS PEMAHAMAN MAHASISWA TERHADAP KONSEP LIMIT FUNGSI DI SATU TITIK (STUDI KASUS PADA MAHASISWA JURUSAN MATEMATIKA FMIPA UNM). Makasar: Pasca Sarjana Universitas Negeri Makasar.

Khasanah, B.A., dkk. (2020). LT GAME 20 SEBAGAI MEDIA PEMBELAJARAN INTERAKTIF PADA MATERI LIMIT TRIGONOMETRI. Lampung: Program Studi Pendidikan Matematika Universitas Muhammadiyah Pringsewu Lampung.