

LAPORAN TUGAS BESAR AKHIR SEMESTER
Struktur Data dan Algoritma

Kalkulator Tree



Disusun oleh

FIORA BERLIANA PUTRI - 201524045
LAMDA RICHO VANJAYA SUMARYADI - 201524049
MUHAMAD ARYADIPURA SASMITA ATMADJA - 201524054

Program Studi D-IV Teknik Informatika
Departemen Teknik Komputer dan Informatika
Politeknik Negeri Bandung
2021

DAFTAR ISI

DAFTAR GAMBAR	ii
DAFTAR TABEL	iii
BAB 1	4
DESKRIPSI APLIKASI	4
A. PENJELASAN APLIKASI	4
B. FITUR APLIKASI	4
BAB 2	5
RANCANGAN APLIKASI	5
A. VISUALISASI PENYELESAIAN PERMASALAHAN	5
B. STRUKTUR DATA DAN ALGORITMA	10
C. RANCANGAN USER INTERFACE	28
BAB 3	31
KONTRIBUSI ANGGOTA DAN TIMELINE Pengerjaan	31
BAB 4	34
SIMULASI	34
A. RANCANGAN ALUR SKENARIO SIMULASI	34
B. DATA SIMULASI	34
BAB 5	38
HASIL AKHIR PROGRAM	38
A. EKSPERIMEN / PENGUJIAN	38
B. LESSON LEARNED	42
DAFTAR PUSTAKA	43

DAFTAR GAMBAR

Gambar 1 Alur Skenario Simulasi Program	34
Gambar 2 Hasil Pengujian 1	38
Gambar 3 Hasil Pengujian 2	38
Gambar 4 Hasil Pengujian 3	38
Gambar 5 Hasil Pengujian 4	39
Gambar 6 Hasil Pengujian Persegi	39
Gambar 7 Hasil Pengujian Persegi Panjang	39
Gambar 8 Hasil Pengujian Segitiga	40
Gambar 9 Hasil Pengujian Trapesium	40
Gambar 10 Hasil Pengujian Jajar Genjang	40
Gambar 11 Hasil Pengujian Belah Ketupat	41
Gambar 12 Hasil Pengujian Layang-Layang	41
Gambar 13 Hasil Pengujian Lingkaran	41

DAFTAR TABEL

Tabel 1 Modul InfixToPostfix	5
Tabel 2 Modul BuildExpressionTree	8
Tabel 3 Modul CalculationOfTree	9
Tabel 4 Daftar Modul ADT Tree	10
Tabel 5 Daftar Modul ADT StackForChar	14
Tabel 6 Daftar Modul ADT StackForTree	16
Tabel 7 Daftar Modul ADT Kalkulator	18
Tabel 8 Daftar Modul ADT Bangun Datar	24
Tabel 9 Kontribusi Fiora Berliana Putri	31
Tabel 10 Kontribusi Lamda Richo Vanjaya Sumaryadi	32
Tabel 11 Kontribusi Muhamad Aryadipura Sasmita Atmadja	33
Tabel 12 Data Simulasi Kalkulator 1	34
Tabel 13 Data Simulasi Kalkulator 2	35
Tabel 14 Data Simulasi Kalkulator 3	35
Tabel 15 Data Simulasi Bangun Datar	37

BAB 1

DESKRIPSI APLIKASI

A. PENJELASAN APLIKASI

Kalkulator atau mesin hitung adalah alat untuk menghitung dari perhitungan sederhana seperti penjumlahan, pengurangan, perkalian, dan pembagian. Kalkulator biasa digunakan untuk menemukan solusi matematika tertentu dengan cepat dan akurat. Selain perhitungan sederhana, kalkulator ini juga dapat mengoperasikan operasi matematika yang kompleks seperti perpangkatan dan pengakaran kuadrat.

B. FITUR APLIKASI

Aplikasi yang akan kami buat adalah aplikasi kalkulator yang menggunakan konsep binary tree dalam melakukan proses perhitungannya. Dalam proses perhitungannya dibutuhkan inputan berupa string yaitu sebuah ekspresi matematika yang mengandung operator dan operand. Operator yang berlaku dalam aplikasi kalkulator ini, diantaranya (), ^, 2v, * atau x, : atau /, +, dan -, sedangkan operand-nya berupa bilangan bulat dan bilangan desimal. Aplikasi ini akan menentukan operator mana yang akan diproses terlebih dahulu berdasarkan derajat operatornya. Selanjutnya memberikan hasil perhitungan ke layar dan output yang dihasilkan akan berupa bilangan desimal dengan dua angka di belakang koma.

Misalnya, user menginputkan :	dengan cara pengerjaan sebagai berikut :
$7+6*5+(2-1)$	$7+6*5+(2-1)$
maka, outputnya :	$= 7+30+(2-1)$
38	$= 7+30+1$
	$= 37+1$
	$= 38$

Rencananya fitur khas yang terdapat dalam aplikasi Kalkulator ini adalah *user* bisa menghitung luas dan keliling pada bangun datar seperti persegi, persegi panjang, lingkaran, dll.

Misalkan *user* memilih rumus persegi panjang. Setelah itu, *user* diminta menginput panjang dan lebarnya dalam ukuran *centimeter* (cm).

Contoh :

Input :

Masukkan panjang	10
Masukkan lebar	5

Output :

Luas	50
Keliling	30

dengan cara pengerjaan sebagai berikut :

- Menghitung luas persegi panjang

$$L = p \cdot l$$

$$L = 10 \cdot 5$$

$$L = 50 \text{ cm}$$

- Menghitung keliling persegi panjang

$$K = 2 \cdot (p + l)$$

$$K = 2 \cdot (10 + 5)$$

$$K = 2 \cdot (15)$$

$$K = 30 \text{ cm}$$

BAB 2

RANCANGAN APLIKASI

A. VISUALISASI PENYELESAIAN PERMASALAHAN

Alur program kami diawali dengan inputan *user*, InfixToPostfix, BuildExpressionTree, dan terakhir CalculationOfTree.

Contoh inputan user : $7+6*5+(2-1)$


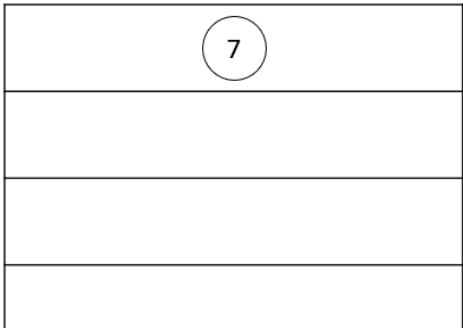
1. Modul InfixToPostfix

Karakter yang dibaca	Isi Stack	Karakter yang tercetak	Postfix yang terbentuk
7		7	7
+	+		
6	+	6	7 6
*	+ *		
5	+ *	5	7 6 5
+	+ +	*	7 6 5 *
((+	+	7 6 5 * +
2		2	7 6 5 * + 2
-	(+ -		
1		1	7 6 5 * + 2 1
)	(+	-	7 6 5 * + 2 1 -
	(+	7 6 5 * + 2 1 - +

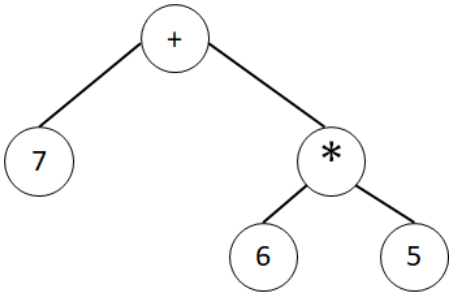
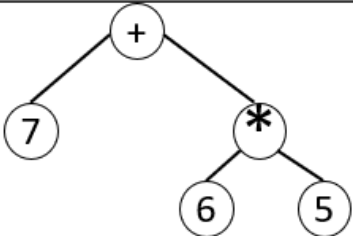

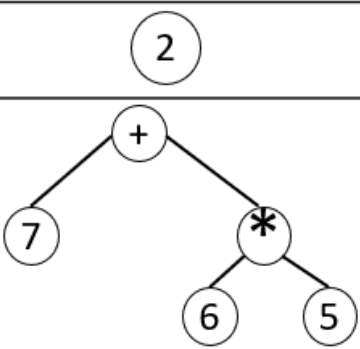

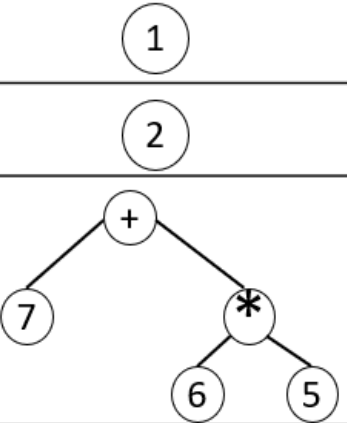
Tabel 1 Modul InfixToPostfix

Postfix yang terbentuk : $7\ 6\ 5\ *\ +\ 2\ 1\ -\ +$

2. Modul BuildExpressionTree

Nilai yang dibaca	Isi ExpressionTree	Isi Stack Tree
7		

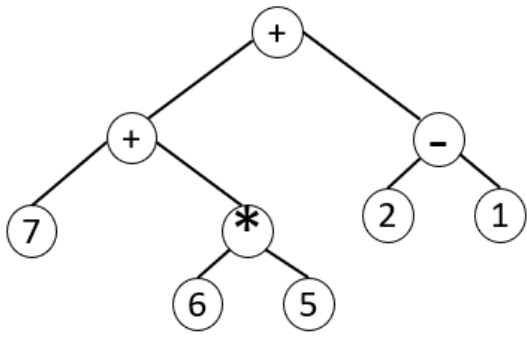
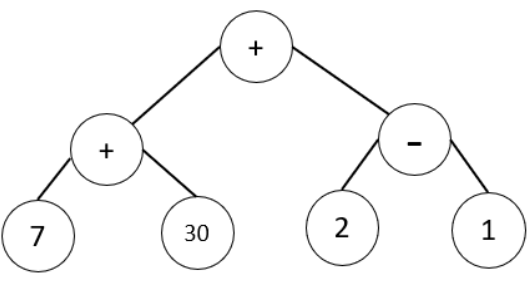
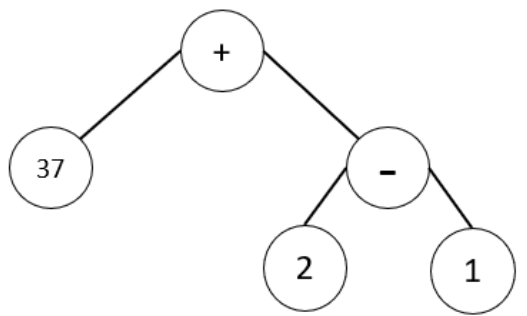
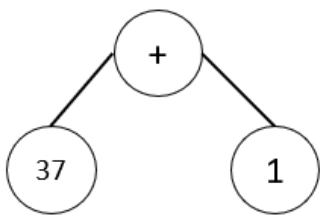
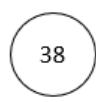
6	<div><div>6</div></div>	<div><div><div>6</div></div><div><div>7</div></div><div></div><div></div></div>
5	<div><div>5</div></div>	<div><div><div>5</div></div><div><div>6</div></div><div><div>7</div></div><div></div></div>
*	<div><div><div>*</div><div>6</div><div>5</div></div></div>	<div><div><div><div>*</div><div>6</div><div>5</div></div><div><div>7</div></div><div></div><div></div></div></div>

+		<div data-bbox="927 208 1378 761">  </div> <div data-bbox="927 454 1378 761"></div>
2		<div data-bbox="927 835 1378 1361">  </div> <div data-bbox="927 1193 1378 1361"></div>
1		<div data-bbox="927 1451 1378 1973">  </div> <div data-bbox="927 1895 1378 1973"></div>

-	<pre> graph TD A((-)) --- B((2)) A --- C((1)) </pre>	<div> <pre> graph TD A((-)) --- B((2)) A --- C((1)) </pre> </div> <div></div> <div></div>
+	<pre> graph TD A((+)) --- B((+)) A --- C((-)) B --- D((7)) B --- E((*)) E --- F((6)) E --- G((5)) C --- H((2)) C --- I((1)) </pre>	<div> <pre> graph TD A((+)) --- B((+)) A --- C((-)) B --- D((7)) B --- E((*)) E --- F((6)) E --- G((5)) C --- H((2)) C --- I((1)) </pre> </div> <div></div> <div></div> <div></div>
	<pre> graph TD A((+)) --- B((+)) A --- C((-)) B --- D((7)) B --- E((*)) E --- F((6)) E --- G((5)) C --- H((2)) C --- I((1)) </pre>	<div></div> <div></div> <div></div> <div></div>

Tabel 2 Modul BuildExpressionTree

3. Modul CalculationOfTree

ExpressionTree	Proses Perhitungan
 <pre> graph TD A((+)) --- B((+)) A --- C((-)) B --- D((7)) B --- E((*)) E --- F((6)) E --- G((5)) C --- H((2)) C --- I((1)) </pre>	<p>left : 6, right 5 $\rightarrow 6*5 = 30$</p>
 <pre> graph TD A((+)) --- B((+)) A --- C((-)) B --- D((7)) B --- E((30)) C --- H((2)) C --- I((1)) </pre>	<p>left : 7, right : 30 $\rightarrow 7 + 30 = 37$</p>
 <pre> graph TD A((+)) --- B((37)) A --- C((-)) C --- H((2)) C --- I((1)) </pre>	<p>left : 2, right : 1 $\rightarrow 2 - 1 = 1$</p>
 <pre> graph TD A((+)) --- B((37)) A --- C((1)) </pre>	<p>left : 37, right : 1 $\rightarrow 38$</p>
 <pre> graph TD A((38)) </pre>	<p>\rightarrow hasil akhir</p>

Tabel 3 Modul CalculationOfTree

B. STRUKTUR DATA DAN ALGORITMA

1. ADT Tree

```
typedef char String[50];
typedef char infotypeTree[10];
typedef struct treeNode *addressTree;
typedef struct treeNode {
    infotypeTree info;
    addressTree left;
    addressTree right;
} treeNode;
typedef addressTree BinTree;
```

Jenis	Nama Modul	Parameter Input	Parameter (I/O)	Keterangan
Function	AlokasiTree	-	infotypeTree X	Menghasilkan address hasil alokasi expression tree
Function	IsEmptyTree	-	BinTree P	Mengembalikan true jika tree kosong
Function	LeftChild	-	BinTree P	Mengembalikan anak kiri
Function	RightChild	-	BinTree P	Mengembalikan anak kanan
Procedure	CreateNodeTree	-	BinTree *P, infotypeTree X	Membuat node tree baru
Procedure	CreateTree	infotypeTree X, BinTree L, BinTree R	BinTree *P	Membuat sebuah expression tree
Procedure	ShowInfoTree	BinTree P	-	Menampilkan semua informasi dari setiap node pada tree
Procedure	StringToFloat	String X	-	Melakukan casting terhadap sebuah string menjadi float.
Function	isOperator	char c	-	Mengembalikan true jika karakter yang diperiksa merupakan operator
Function	Priority	char x	-	Mengembalikan nilai prioritas dari sebuah operator (semakin besar nilai, prioritas semakin diutamakan)
Function	isPriority	char a, char b	-	mengembalikan true jika prioritas operator1 lebih tinggi dari prioritas operator2

Tabel 4 Daftar Modul ADT Tree

Algoritma :

1. Function

```
AlokasiTree (X : infotypeTree)
//Kamus Data
P : addressTree

//Algoritma
P <- (addressTree)alloc(sizeof(treeNode))

if (P <> Nil) then
    Info(P) <- X
    Left(P) <- Nil
    Right (P) <- Nil
endif

-> P
end function
```

2. Function

```
IsEmptyTree(P : BinTree)
-> (P <- Nil)
End
```

3. Function

```
LeftChild (P : BinTree)
-> Left(P)
End
```

4. Function

```
RightChild (P : BinTree)
-> Right(P)
End
```

5. Procedure

```
CreateNodeTree(*P : BinTree, X : infotypeTree)
    *P <- AlokasiTree(X)
    if(*P <> Nil) then
        Info(*P) <- X
        Left (*P) <- Nil
        Right (*P) <- Nil
    endif
end
```

6. Procedure

```
CreateTree(X : infotypeTree, L : BinTree, R : BinTree, *p :
BinTree)
    *p <- AlokasiTree(X)
    if(*p <> Nil) then
```

```

    Info(*P) <- X
    Left(*P) <- L
    Right(*P) <- R
endif
end function

```

7. Procedure

```

ShowInfoTree(P : BinTree)
//Kamus Data
L : BinTree
R : BinTree
x : BinTree
//Algoritma
if(P<>Nil) then
    ShowInfoTree(Left(P))
    Write(Layar)<- "Info Node\t: %s\n", Info(P)

    if(LeftChild(P) <> Nil) then
        L = LeftChild(P)
        write(layar) <- "Left Child\t: %s\n", Info(L)
    else
        write(layar) <- "Left Child\t: NULL"
    end if

    if(RightChild(P)<>Nil) then
        R = RightChild(P)
        write(layar) <- "Right Child\t: %s\n", RightChild(P)
    else
        write(layar) <- "Right Child\t: NULL"
    end if

    write(layar) <- "\n"
    ShowInfoTree(Right(P))

end procedure

```

8. Procedure

```

StringToFloat(String X)
-> atof(X)//fungsi untuk mengonversi string menjadi int
end procedure

```

9. Procedure

```

isOperator(char c)
-> (c='+' OR c='-' OR c='*' OR c='x' OR c='/' OR c=':' OR c='^'
OR c='v')
end procedure

```

10. Function

```

priority(char x) do
    switch(x) do

```

```

case ')' : -> 0
case '(' : -> 0
case '+' : -> 1
case '-' : -> 1
case '*' : -> 2
case 'x' : -> 2
case '/' : -> 2
case ':' : -> 2
case '^' : -> 3
case 'v' : -> 3
end switch
end function

```

11. Function

```

isPriority(a : char, b : char) do
-> (Priority(a)>=Priority(b))
end function

```

2. ADT Stack

a. ADT StackForChar

```

typedef char infotypeStackChar
typedef struct tElmStackChar *addressStackChar;
typedef struct tElmStackChar {
    infotypeStackChar info;
    addressStackChar next;
}ElmStackChar;

typedef struct{
    addressStackChar top;
}StackChar;

```

Jenis	Nama Modul	Parameter Input	Parameter (I/O)	Keterangan
Function	alokasiStackChar	-	infotypeStackChar X	Mengirim sebuah elemen stack char dalam bentuk address
Procedure	dealokasiStackChar	-	addressStackChar P	Membebaskan/ Melepaskan address P
Function	IsEmptyStackChar	-	StackChar S	Mengembalikan true jika stack char kosong
Procedure	NewStackChar	-	StackChar *S	Membuat sebuah stack char kosong
Procedure	AddStackChar	-	StackChar *S, infotypeStackChar X	Melakukan push elemen ke bagian top stack char

Procedure	DellStackChar	-	StackChar *S, infotypeStackChar *X	Menghapus elemen dari bagian top stack char
-----------	---------------	---	---------------------------------------	---

Tabel 5 Daftar Modul ADT StackForChar

Algoritma

1. Function

```

alokasiStackChar(X : infotypeStackChar)

//Kamus Data
P : addressStackChar

//Algoritma
Begin
    P <- (addressStackChar) alloc(sizeof(elmStackChar))
    if (P<>Nil) then
        Info(P) <- X
        Next(P) <- Nil
    End if
    -> P
End function

```

2. Procedure

```

dealokasiStackChar(P : addressStackChar)
//Algoritma
Begin
    free (P)
End function

```

3. Function

```

isEmptyStackChar(S : stackChar)
//Algoritma
Begin
    -> (Top(S) == Nil)
End function

```

4. Procedure

```

NewStackChar(S : stackChar)
//Algoritma
Begin
    Top(S) <- Nil
End procedure

```

5. Procedure

```

DellStackChar(*S : stackChar, *X : infotypeStackChar)
//Kamus Data
P : addressStackChar

```

```

//Algoritma
Begin
    if (!isEmptyStackChar(*S)) then
        P <- Top(*S)
        Top(*S) <- Next(P)
        *X <- Info(P)
        dealokasiStackChar(P)
    else
        write(layar) <- "StackChar kosong!"
    End if
End procedure

```

6. Procedure

```

AddStackChar(*S : stackChar, X : infotypeStackChar)
//Kamus Data
P : addressStackChar

//Algoritma
Begin
    P <- alokasiStackChar(X)
    Next(P) <- Top(*S)
    Top(*S) <- P
End procedure

```

b. ADT StackForTree

```

typedef BinTree infotypeStackTree;
typedef struct tElmSackTree *addressStackTree;
typedef struct tElmStackTree{
    infotypeStackTree info;
    addressStackTree next;
}ElmStackTree;

typedef struct{
    addressStackTree top;
}StackTree;

```

Jenis	Nama Modul	Parameter Input	Parameter (I/O)	Keterangan
Function	alokasiStackTree	-	infotypeStackTree X	Mengirim sebuah elemen stack tree dalam bentuk address
Procedure	dealokasiStackTree	-	addressStackTree P	Membebaskan/ Melepaskan address P
Function	isEmptyStackTree	-	StackTree S	Mengembalikan true jika tree kosong
Procedure	NewStackTree	-	StackTree *S	Membuat sebuah stack tree kosong

Procedure	AddStackTree	-	StackTree *S, infotypeStackTree X	Melakukan push elemen ke bagian top stack Tree
Procedure	DellStackTree	-	StackTree *S, infotypeStackTree *X	Menghapus elemen dari bagian top stack tree

Tabel 6 Daftar Modul ADT StackForTree

Algoritma :

1. Function

```

alokasiStackTree(X : infotypeStackTree)
//Kamus Data
P : addressStackTree
//Algoritma
Begin
    P <- (addressStackTree) alloc(sizeof(ElmStackTree))
    if(P<>Nil) then
        Info(P)<- X
        Next(P)<- Nil
    End if
    -> P
End function

```

2. Procedure

```

dealokasiStackTree(P : addressStackTree)
//Algoritma
Begin
    free (P)
End procedure

```

3. Function

```

isEmptyStackTree(S : stackTree)
//Algoritma
Begin
    -> (Top(S)<- Nil)
End function

```

4. Procedure

```

NewStackTree(*S : stackTree)
//Algoritma
Begin
    Top(S)<- Nil
End procedure

```

5. Procedure

```

DellStackTree(*S : stackTree, *X : infotypeStackTree)
//Kamus Data

```

```

P : addressStackTree
//Algoritma
Begin
    if (!isEmptyStackTree(*S)) then
        P <-Top(*S)
        Top(*S)<-Next(P)
        *X <-Info(P)
        dealokasiStackTree(P)
    else
        write(layar)<-"StackTree kosong!"
    endif
End procedure

```

6. Procedure

```

AddStackTree(*S : stackTree, X : infotypeStackTree)
//Kamus Data
addressStackTree P;
//Algoritma
Begin
    P <-alokasiStackTree(X)
    Next(P)<-Top(*S)
    Top(*S)<-P
End procedure

```

3. ADT Kalkulator

```

typedef char String[50];
typedef char infotypetree[10];
typedef struct treeNode *addressTree;
typedef addressTree Kalkulator;

```

Jenis	Nama Modul	Parameter Input	Parameter (I/O)	Keterangan
Function	BuildExpressionTree	InfotypeTree postfix	-	Membuat expression tree dari postfix
Function	CalculationOfTree	BinTree P	-	Mengembalikan kalkulasi dari ekspresi sebuah tree
Procedure	InfixToPostfix	String Infix	String Postfix	Mengonversi ekspresi infix ke ekspresi postfix
Procedure	MenuKalkulator	-	-	Menampilkan menu untuk kalkulator
Procedure	MenuBangunDatar	-	-	Menampilkan menu untuk bangun datar
Procedure	MenuAbout	-	-	Menampilkan informasi mengenai developer aplikasi kalkulator

Function	MainMenu	-	-	Menampilkan menu utama dari aplikasi kalkulator
----------	----------	---	---	---

Tabel 7 Daftar Modul ADT Kalkulator

Algoritma

1. Function

```

BuildExpressionTree(postfix : infotypeTree)
//Kamus Data
i <- 0, j, k : integer
StackTree : StackTree
ExpressionTree, Right, Left : BinTree
Delete : infotypeStackTree
tempOperator, tempStr : infotypeTree

//Algoritma
NewStackTree(&StackTree)

while (i<strlen(postfix)) do
    if (!isOperator(postfix[i]) and postfix[i]<> ' ') then
        j <- 0
        tempStr[j] <- postfix[i]
        while (!isOperator(postfix[i+1]) and postfix[i+1]!=' ') do
            tempStr[j+1] <- postfix[i+1]
            j++
            i++
        endWhile

        CreateNodeTree(&ExpTree, tempStr)
        for (k = 0, k < strlen(tempStr), k++)
            tempStr[k] <- ' '
        endFor
        AddStackTree(&StackTree, ExpressionTree)
    endIf

    else if (postfix[i]=='-' and isOperator(postfix[i-4])) then
        tempOperator[0] <- postfix[i]
        CreateNodeTree(&ExpTree, tempOperator)

        Right <- Info(Top(StackTree))
        DellStackTree(&StackTree, &Delete)

        Left <- Info(Top(StackTree))
        DellStackTree(&StackTree, &Delete)

        MakeTree(tempOperator, Left, Right, &ExpressionTree)

        AddStackTree(&StackTree, ExpressionTree)
    endIf

    else if (
        (postfix[i]=='-' and postfix[i-1]=' ' and postfix[i+1]<>' ' and
        postfix[i+2]<>' ' and !isOperator(postfix[i+2])) or
        (postfix[i]=='-' and postfix[i-1]<>' ' and postfix[i+1]=' ') or

```

```

(postfix[i]='-' and postfix[i-1]<>' ' and
isOperator(postfix[i+1])) or (postfix[i]='-' and postfix[i-1]=' '
and postfix[i+1]=' ') or
(postfix[i]='-' && postfix[i-1]=' ' and postfix[i+1]<>' ' and
isOperator(postfix[i+2])) or (postfix[i]='-' && postfix[i-1]<>' '
and isOperator(postfix[i+1]) and isOperator(postfix[i+2])) )
    then
        tempOperator[0]<- postfix[i]
        CreateNodeTree(&ExpressionTree, tempOperator)

        Right <- Info(Top(StackTree))
        DellStackTree(&StackTree, &Delete)

        Left <- Info(Top(StackTree))
        DellStackTree(&StackTree, &Delete)

        MakeTree(tempOperator, Left, Right, &ExpressionTree)

        AddStackTree(&StackTree, ExpressionTree)
    endIf

    else if (postfix[i]<>' ') then
        tempOperator[0]<-postfix[i]
        CreateNodeTree(&ExpressionTree, tempOperator)

        Right <- Info(Top(StackTree))
        DellStackTree(&StackTree, &Delete)

        Left <- Info(Top(StackTree))
        DellStackTree(&StackTree, &Delete)

        MakeTree(tempOperator, Left, Right, &ExpressionTree)

        AddStackTree(&StackTree, ExpressionTree)
    endIf
    i++
endWhile
ExpressionTree <- Info(Top(StackTree))
DellStackTree(&StackTree, &ExpressionTree)
-> ExpressionTree

end function

```

2. Function

```

CalculationOftree(P : BinTree)
//Algoritma
if (IsEmptyTree(P)) then
    return 0;
endIf
else if (IsEmptyTree(Left(P)) AND IsEmptyTree(Right(P))) then
    r-> StringToFloat(Info(P))
endIf

```

```

float left <- CalculationOfTree(Left(P))
float right <- CalculationOfTree(Right(P))

if(Info(P) = "+") then
  -> left+right

else if(Info(P) = "-") then
  -> left-right

else if(Info(P) = "*") then
  -> left*right
else if(Info(P) = "x") then
  -> left*right

else if(Info(P) = "/") then
  if(right <> 0.00){
    -> left/right
  } else{
    puts("Tidak bisa dibagi oleh nol")
  }
else if(Info(P) = ":") then
  if(right <> 0.00){
    -> left/right
  } else{
    puts("Tidak bisa dibagi oleh nol")
  }
else if(Info(P) = "^") then
  -> pow(left,right)

else if(Info(P) = "v") then
  -> sqrt(right)

endIf
end function

```

3. Procedure

```

InfixToPostfix(infix : String, postfix : String)
//Kamus Data
i : integer
index : integer
index <- 0
ukuran: integer
tempChar : char
temp : stackChar

NewStackChar(&temp)

size <- strlen(infix)
for(i=0 to ukuran;i++)
  switch(infix[i])
    case \'.' :
    case \'0\' :
    case \'1\' :

```

```

case '2' :
case '3' :
case '4' :
case '5' :
case '6' :
case '7' :
case '8' :
case '9' :
    postfix[index] <- infix[i]
    index <- index + 1
    break
case '+' :
case '-' :
case '*' :
case 'x' :
case '/' :
case ':' :
case '^' :
case 'v' :
    postfix[index] <- ' '
    index <- index + 1

    if(isEmptyStackChar(temp)) then
        pushStackChar(&temp, infix[i])

    else if(!isPriority(Info(Top(temp)),infix[i])) then
        AddStackChar(&temp, infix[i])

    else
        while(!isEmptyStackChar(temp) AND
            isPriority(Info(Top(temp)), infix[i])) do
            DellStackChar(&temp,&tempChar)
            postfix[index] <- tempChar
            index <- index + 1
        endwhile
        AddStackChar(&temp,infix[i])
    endif
    break;
case ')' :
    while(!IsEmptyStackChar(temp) AND Info(Top(temp))!='(') do
        DellStackChar(&temp, &tempchar)
        postfix[index] <- tempChar
        index <- index + 1
    endwhile
    DellStackChar(&temp,&tempChar)
    break;
case '(' :
    AddStackChar(&temp,infix[i])
    break
endSwitch
endFor
while(!IsEmptyStackChar(temp)) do
    DellStackChar(&temp, &tempChar)
    postfix[index] <- tempChar
    index <- index + 1

```

```

endWhile
postfix[index]='\0'
end procedure

```

4. ADT Bangun Datar

```

typedef struct{
    int sisi;
}Persegi;

typedef struct{
    int panjang;
    int lebar;
}PersegiPanjang;

```

```

typedef struct{
    int alas;
    int tinggi;
    int sisi;
}Segitiga;

typedef struct{
    int sejajar1;
    int sejajar2;
    int sisi1;
    int sisi2;
    int tinggi;
}Trapeسيوم;

```

```

typedef struct{
    int alas;
    int tinggi;
    int sisi;
}JajarGenjang;

typedef struct{
    int diagonal1;
    int diagonal2;
    int sisi;
}BelahKetupat;

```

```

typedef struct{
    int diagonal1;
    int diagonal2;
    int sisi1;
    int sisi2;
}LayangLayang;

typedef struct{
    int jariJari;
}Lingkaran;

```

Jenis	Nama Modul	Parameter Input	Keterangan
Function	LuasPersegi	Persegi P	Mengembalikan hasil dari luas persegi berdasarkan inputan user
Function	KelilingPersegi	Persegi P	Mengembalikan hasil dari keliling persegi berdasarkan inputan user
Procedure	HitungPersegi	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.
Function	LuasPersegiPanjang	PersegiPanjang P	Mengembalikan hasil dari luas persegi panjang berdasarkan inputan user
Function	KelilingPersegiPanjang	PersegiPanjang P	Mengembalikan hasil dari keliling persegi panjang berdasarkan inputan user
Procedure	HitungPersegiPanja	-	Tempat user untuk menginputkan nilai.

	ng		Lalu akan menampilkan hasil luas dan kelilingnya.
Function	LuasSegitiga	Segitiga P	Mengembalikan hasil dari luas segitiga berdasarkan inputan user
Function	KelilingSegitiga	Segitiga P	Mengembalikan hasil dari keliling segitiga berdasarkan inputan user
Procedure	HitungSegitiga	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.
Function	LuasTrapeسيوم	Trapeسيوم P	Mengembalikan hasil dari luas trapeسيوم berdasarkan inputan user
Function	KelilingTapeسيوم	Trapeسيوم P	Mengembalikan hasil dari keliling trapeسيوم berdasarkan inputan user
Procedure	HitungTrapeسيوم	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.
Function	LuasJajarGenjang	JajarGenjang P	Mengembalikan hasil dari luas jajar genjang berdasarkan inputan user
Function	KelilingJajarGenjang	JajarGenjang P	Mengembalikan hasil dari keliling jajar genjang berdasarkan inputan user
Procedure	HitungJajarGenjang	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.
Function	LuasBelahKetupat	BelahKetupat P	Mengembalikan hasil dari luas belah ketupat berdasarkan inputan user
Function	KelilingBelahKetupat	BelahKetupat P	Mengembalikan hasil dari keliling belah ketupat berdasarkan inputan user
Procedure	HitungBelahKetupat	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.
Function	LuasLayangLayang	LayangLayang P	Mengembalikan hasil dari luas layang-layang berdasarkan inputan user
Function	KelilingLayangLayang	LayangLayang P	Mengembalikan hasil dari keliling layang-layang berdasarkan inputan user
Procedure	HitungLayangLayang	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.

Function	LuasLingkaran	Lingkaran P	Mengembalikan hasil dari luas lingkaran berdasarkan inputan user
Function	KelilingLingkaran	Lingkaran P	Mengembalikan hasil dari keliling lingkaran berdasarkan inputan user
Procedure	HitungLingkaran	-	Tempat user untuk menginputkan nilai. Lalu akan menampilkan hasil luas dan kelilingnya.

Tabel 8 Daftar Modul ADT Bangun Datar

Algoritma :

1. Function

```
LuasPersegi(P : Persegi)
-> Sisi(P)*Sisi(P)
end function
```

2. Function

```
KelilingPersegi(P : Persegi)
-> 4*Sisi(P)
end function
```

3. Procedure

```
HitungPersegi(P : Persegi)
Persegi P;
write(layar)<-( "\n\t\t Hitung Persegi \n")
write(layar)<-( "\nMasukkan Sisi\t : ")
read(keyboard)<-( "%d", &Sisi(P))
write(layar)<-( "\nLuas Persegi\t = %.2f", LuasPersegi(P))
write(layar)<-( "\nKeliling Persegi = %.2f\n",
KelilingPersegi(P))
end procedure
```

4. Function

```
LuasPersegiPanjang(P : PersegiPanjang)
-> 2*(Panjang(P)+Lebar(P))
end function
```

5. Function

```
KelilingPersegiPanjang(P : PersegiPanjang)
-> 2*(Panjang(P)+Lebar(P))
end function
```

6. Procedure

```
HitungPersegiPanjang()  
  P : PersegiPanjang  
  write(layar)<-"\n\t\t Hitung Persegi Panjang \n"  
  write(layar)<-"Masukkan Panjang\t : "  
  read(keyboard)<- "%d", &Panjang(P)  
  
  write(layar)<-"Masukkan Lebar\t\t : "  
  read(keyboard)<- "%d", &Lebar(P)  
  
  write(Layar)<-"Luas Persegi Panjang\t = %.2f",  
  LuasPersegiPanjang(P)  
  
  write(layar)<-"Keliling Persegi Panjang = %.2f\n",  
  KelilingPersegiPanjang(P)  
end procedure
```

7. Function

```
LuasSegitiga(P : Segitiga)  
  -> 0.5*Alas(P)*Tinggi(P)  
end function
```

8. Function

```
KelilingSegitiga(P : Segitiga)  
  -> 3*Alas(P)  
end function
```

9. Function

```
HitungSegitiga()  
  P : Segitiga  
  write(layar)<- "\n\t\t Hitung Segitiga \n"  
  write(layar)<- "\nMasukkan Alas\t\t : "  
  write(layar)<- "%d", &Alas(P)  
  write(layar)<- "Masukkan Tinggi\t\t : "  
  read(keyboard)<- "%d", &Tinggi(P)  
  write(layar)<- "Masukkan Sisi Miring\t : "  
  read(keyboard)<- "%d", &Sisi(P)  
  write(layar)<- "\nLuas Segitiga\t\t = %.2f", LuasSegitiga(P)  
  write(layar)<- "\nKeliling Segitiga\t = %.2f\n",  
  KelilingSegitiga(P)  
end function
```

10. Function

```
LuasTrapeسيوم(P : Trapesium)  
  -> 0.5*(RusukSejajar1(P)RusukSejajar2(P))*Tinggi(P)  
end function
```

11. Function

```
KelilingTrapeسيوم(P : Trapezium)
-> Sisi1(P)+Sisi2(P)+RusukSejajar1(P)+RusukSejajar2(P)
end function
```

12. Procedure

```
HitungTrapeسيوم()
  P : Trapezium
  write(layar)<-"\\n\\t\\t Hitung Trapezium \\n"
  write(layar)<-"\\nMasukkan Rusuk Sejajar 1\\t : "
  read(keyboard)<-"%d", &RusukSejajar1(P)
  write(layar)<-"Masukkan Rusuk Sejajar 2\\t : " read(keyboard)<-"%d", &RusukSejajar2(P)
  write(layar)<-"Masukkan Tinggi\\t\\t\\t : " read(keyboard)<-"%d", &Tinggi(P)
  write(layar)<-"Masukkan Sisi Kiri\\t\\t : " read(keyboard)<-"%d", &Sisi1(P)
  write(layar)<-"Masukkan Sisi Kanan\\t\\t : " read(keyboard)<-"%d", &Sisi2(P)
  write(layar)<-"\\nLuas Trapezium\\t\\t = %.2f", LuasTrapeسيوم(P)
  write(layar)<-"\\nKeliling Trapezium\\t = %.2f\\n",
KelilingTrapeسيوم(P)
end procedure
```

13. Function

```
LuasJajarGenjang(P : JajarGenjang)
-> Alas(P)*Tinggi(P)
end function
```

14. Function

```
KelilingJajarGenjang(P : JajarGenjang)
-> 2*(Sisi(P)+Alas(P))
end function
```

15. Procedure

```
HitungJajarGenjang()
  P : JajarGenjang
  write(layar)<-"\\n\\t\\t Hitung Jajar Genjang \\n"
  write(layar)<-"\\nMasukkan Alas\\t\\t : "
  read(keyboard)<-"%d", &Tinggi(P)
  write(layar)<-"Masukkan Sisi Miring\\t : "
  read(keyboard)<-"%d", &Sisi(P)
  write(layar)<-"\\nLuas Jajar Genjang\\t = %.2f", LuasJajarGenjang(P)
  write(layar)<-"\\nKeliling\\t\\t = %.2f\\n", KelilingJajarGenjang(P)
end procedure
```

16. Function

```
LuasBelahKetupat(P : BelahKetupat)
-> 0.5*(Diagonal1(P)*Diagonal2(P))
end function
```

17. function

```
KelilingBelahKetupat(P : BelahKetupat)
-> 4*Sisi(P)
end function
```

18. Procedure

```
HitungBelahKetupat()
P : BelahKetupat
write(layer)<-"\\n\\t\\t Hitung Belah Ketupat \\n"
write(layer)<-"\\nMasukkan Diagonal 1\\t : "
read(keyboard)<-"%d", &Diagonal1(P)
write(layer)<-"Masukkan Diagonal 2\\t : "
read(keyboard)<-"%d", &Diagonal2(P)
write(layer)<-"Masukkan Sisi\\t\\t : "
read(keyboard)<-"%d", &Sisi(P)
write(layer)<-"\\nLuas Belah Ketupat \\t = %.2f", LuasBelahKetupat(P)
write(layer)<-"\\nKeliling Belah ketupat    = %.2f\\n"
KelilingBelahKetupat(P)
end procedure
```

19. Function

```
LuasLayangLayang(P : LayangLayang)
-> 0.5*(Diagonal1(P)*Diagonal2(P))
end function
```

20. Function

```
KelilingLayangLayang(P : LayangLayang)
-> 2*(Sisi1(P)+Sisi2(P))
end function
```

21. Procedure

```
HitungLayangLayang()
P : LayangLayang
write(layer)<-"\\n\\t\\t Hitung Layang-Layang \\n"
write(layer)<-"\\nMasukkan Diagonal 1\\t : " read(keyboard)<-"%d", &Diagonal1(P)
write(layer)<-"Masukkan Diagonal 2\\t : " read(keyboard)<-"%d", &Diagonal2(P)
write(layer)<-"Masukkan Sisi Miring Kiri : "
read(keyboard)<-"%d", &Sisi1(P)
write(layer)<-"Masukkan Sisi Miring Kanan : "
read(keyboard)<-"%d", &Sisi2(P)
```

```

        write(layar)<-"\\nLuas Layang-layang\\t= %.2f",
LuasLayangLayang(P)
        write(layar)<-"\\nKeliling Layang-layang\\t= %.2f\\n",
KelilingLayangLayang(P)
end procedure

```

22. Function

```

LuasLingkaran(P : Lingkaran)
-> 3.14*JariJari(P)*JariJari(P)
end function

```

23. Function

```

KelilingLingkaran(P : Lingkaran P)
-> 2*3.14*JariJari(P)
end function

```

24. Procedure

```

HitungLingkaran()
P : Lingkaran
write(layar)<-"\\n\\t\\t Hitung Lingkaran #\\n"
write(layar)<-"\\nMasukkan Jari-Jari\\t: "
read(keyboard)<-"%d", &JariJari(P)
write(layar)<-"\\nLuas Lingkaran\\t\\t= %.2f", LuasLingkaran(P)
write(layar)<-"\\nKeliling Lingkaran\\t= %.2f\\n",
KelilingLingkaran(P)
end procedure

```

C. RANCANGAN USER INTERFACE

1. Tampilan Awal Aplikasi Kalkulator

Aplikasi Kalkulator

Main Menu

1.Kalkulator
2.Hitung Bangun Datar
3.About
4.Keluar

Masukkan pilihan Anda :

2. Tampilan Menu Kalkulator

Saat user memilih angka 1 maka akan muncul tampilan seperti gambar dibawah ini :

Aplikasi Kalkulator

Petunjuk :

1. Gunakan '^' untuk melakukan operasi perpangkatan.
2. Gunakan '*' atau 'x' untuk melakukan operasi perkalian.
3. Gunakan ':' atau '/' untuk melakukan operasi pembagian.
4. Gunakan '+' untuk melakukan operasi penjumlahan.
5. Gunakan '-' untuk melakukan operasi pengurangan.
6. Gunakan '2v' untuk melakukan operasi akar pangkat 2.
7. Bisa menambahkan '(' dan ')' ke dalam operasi perhitungan.
8. Bilangan yang berlaku adalah bilangan bulat dan bilangan desimal.
9. Dilarang untuk menggunakan spasi.

Press any key to continue ...

Setelah user mengklik tombol apa saja untuk melanjutkan program, user dapat memasukkan angka yang akan dihitung, dan akan muncul hasil dari perhitungan tersebut. Tampilannya seperti ini :

Aplikasi Kalkulator

Lakukan Perhitungan:

$7+6*5+(2-1)$

38.00

Press any key to continue...

3. Tampilan Menu Hitung Bangun Datar

Hitung Bangun Datar

Bangun datar yang tersedia :

1. Persegi
2. Persegi Panjang
3. Segitiga
4. Trapesium
5. Jajar Genjang
6. Belah Ketupat
7. Layang - layang
8. Lingkaran

Masukkan pilihan Anda :

Misalnya, user memilih angka 2, maka tampilan akan seperti dibawah ini :

```

                                Hitung Persegi Panjang

Masukkan Panjang : 12
Masukkan Lebar : 3

Luas Persegi Panjang      = 36.00
Keliling Persegi Panjang = 30.00

Press any key to continue...
```

4. Tampilan Menu About

```

                                About Us

Aplikasi Kalkulator ini dibuat oleh :
Fiora Berliana Putri - 201524045
Lamda Richo Vanjaya Sumaryadi - 201524049
Muhamad Aryadipura Sasmita Atmadja - 201524054
Untuk memenuhi Tugas Besar mata kuliah Struktur Data dan
Algoritma.

Press any key to continue...
```

BAB 3

KONTRIBUSI ANGGOTA DAN TIMELINE Pengerjaan

1. Fiora Berliana Putri - 201524045

No	Kontribusi	Catatan	Referensi	Tanggal
1.	Membuat rancangan ADT Kalkulator	Pembuatan rancangan untuk ADT Kalkulator sebagai ADT untuk menghitung inputan user.	http://chandra-mahardika.blogspot.com/2015/10/materi-prefix-infix-dan-postfix.html , https://github.com/ChACiooh/Binary-Tree-Calculator/blob/master/main.c	19-23 Juli 2021
2.	Membuat ADT Bangun Datar	Pembuatan ADT Bangun Datar sebagai fitur khas yang aplikasi kami miliki.		23 Juli 2021
3.	Membuat ADT Kalkulator	Mengimplementasikan rancangan yang sudah dibuat sebelumnya dari ADT Kalkulator.	https://github.com/ChACiooh/Binary-Tree-Calculator/blob/master/main.c	24 Juli 2021
4.	Penyatuan ADT Stack, ADT Tree, dan ADT Kalkulator	Membuat project pada DEV C++		25 Juli 2021
5.	Memperbaiki beberapa bug dan error			25 dan 31 Juli 2021

Tabel 9 Kontribusi Fiora Berliana Putri

2. Lamda Richo Vanjaya Sumaryadi - 201524049

No	Kontribusi	Catatan	Referensi	Tanggal
1.	Membuat ADT Tree	Pembuatan rancangan ADT Tree untuk pembuatan tree	https://github.com/ChACiooh/Binary-Tree-Calculator/blob/master/Tree.h	17-21 Juli 2021

2.	Membuat Algoritma Pseudocode ADT Tree	Pembuatan rancangan algoritma pseudocode sebelum di translate ke bahasa C	https://isallab.com/article/pengertian-dan-contoh-pseudocode-algoritma-tree-struktur-data-part1/#4	22 Juli 2021
3.	Membuat Implementasi File Header dan Main Driver Binary Tree		https://github.com/ChACiooh/Binary-Tree-Calculator/blob/master/Tree.c	22 Juli 2021
4.	Memodifikasi dan Memperbaiki Bug di Semua ADT	Memperbaiki syntax yang salah dan mensinkronisasi penamaan variabel dan method		25 Juli 2021
5.	Memodifikasi Algoritma Pseudocode di Semua ADT	Menyesuaikan algoritma pseudocode berdasarkan code bahasa C yang sudah dimodif		25 Juli 2021

Tabel 10 Kontribusi Lamda Richo Vanjaya Sumaryadi

3. Muhamad Aryadipura Sasmita Atmadja - 201524054

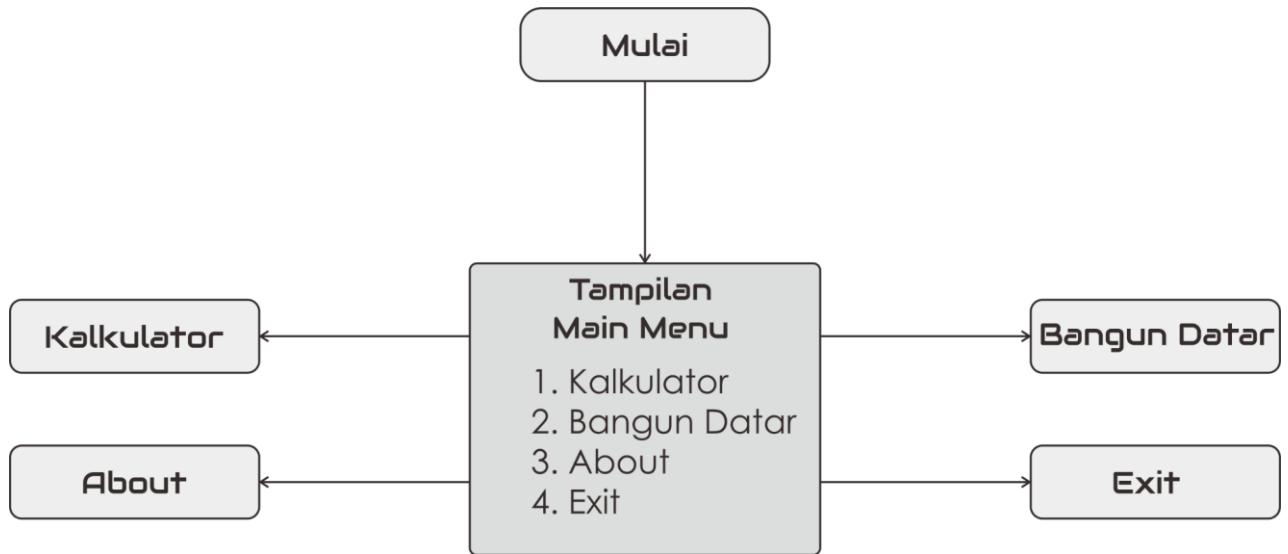
No	Kontribusi	Catatan	Referensi	Tanggal
1.	Membuat ADT Stack	Pembuatan rancangan ADT Stack untuk menampung inputan user	Binary-Tree-Calculator/Stack.h at master · ChACiooh/Binary-Tree-Calculator (github.com)	19-22 juli 2021
2.	Membuat pseudocode ADT StackForChar dan ADT StackForTree			23 Juli 2021
3.	Membuat Implementasi File Header dan Body dari		Binary-Tree-Calculator/Stack.c at master ·	24 Juli 2021

	StackForChar dan StackForTree		ChACiooh/Binary-Tree-Calculator (github.com)	
4.	Melakukan test case untuk mengecek kemungkinan terjadinya bug pada program			25-26 Juli 2021

Tabel 11 Kontribusi Muhamad Aryadipura Sasmita Atmadja

BAB 4 SIMULASI

A. RANCANGAN ALUR SKENARIO SIMULASI



Gambar 1 Alur Skenario Simulasi Program

B. DATA SIMULASI

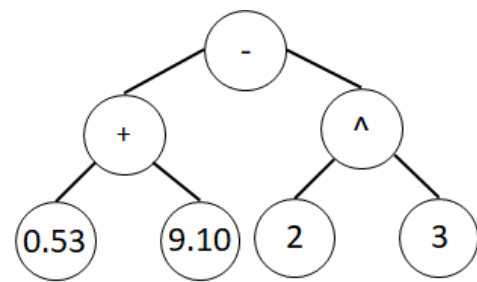
1. Data Simulasi Kalkulator

- Inputan user : 2^4-5

InfixToPostfix :	$2\ 4\ ^\wedge\ 5\ -$
BuildExpressionTree :	<pre> graph TD Minus((-)) --- Caret((^)) Minus --- 5((5)) Caret --- 2((2)) Caret --- 4((4)) </pre>
CalculationOfTree :	1. $2\ ^\wedge\ 4 = 16$ 2. $16 - 5 = 11$ Hasil akhir : 11

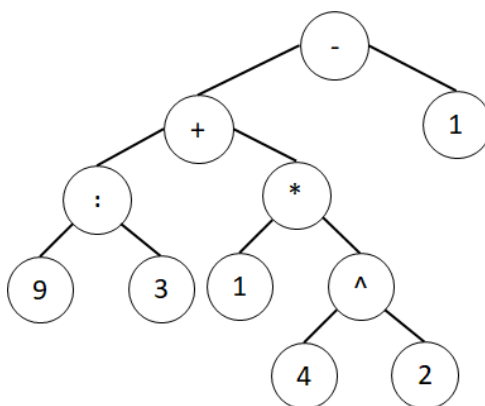
Tabel 12 Data Simulasi Kalkulator 1

- Inputan user : $(0.53+9.10)-2^3$

InfixToPostfix :	0.53 9.10 + 2 3 ^ -
BuildExpressionTree :	
CalculationOfTree :	1. $0.53 + 9.10 = 9.63$ 2. $2^3 = 8$ 3. $9.63 - 8 = 1.63$ Hasil akhir : 1.63

Tabel 13 Data Simulasi Kalkulator 2

- Inputan user : $9:3+(1*(4^2))-1$

InfixToPostfix :	9 3 : 1 4 2 ^ * + 1 -
BuildExpressionTree :	
CalculationOfTree :	1. $4^2 = 16$ 2. $9 : 3 = 3$ 3. $3 + 16 = 19$ 4. $19 - 1 = 18$ Hasil akhir : 18

Tabel 14 Data Simulasi Kalkulator 3

- Inputan user : $((2^4)*(2 \vee 49))$

InfixToPostfix :	$2\ 4\ ^\wedge\ 2\ 49\ \vee\ *$
BuildExpressionTree :	<pre> graph TD A((*)) --- B((^)) A --- C((v)) B --- D((2)) B --- E((4)) C --- F((2)) C --- G((49)) </pre>
CalculationOfTree :	1. $2^\wedge 4 = 16$ 2. $2 \vee 49 = 7$ Hasil akhir : 112

2. Data Simulasi Bangun Datar

No.	Bangun Datar	Input	Proses Perhitungan dan Output
1.	Persegi	Sisi : 4	Keliling : $4 \times s$ $\rightarrow 4 \times 4 = 16$ Luas : $s \times s$ $\rightarrow 4 \times 4 = 16$
2.	Persegi Panjang	Panjang : 4 Lebar : 2	Keliling : $2 \times (p + l)$ $\rightarrow 2 \times (4 + 2) = 12$ Luas : $p \times l$ $\rightarrow 4 \times 2 = 8$
3.	Segitiga	Alas : 6 Tinggi : 5	Keliling : $3 \times s$ $\rightarrow 3 \times 6 = 18$ Luas : $0.5 \times a \times t$ $\rightarrow 0.5 \times 6 \times 5 = 15$
4.	Trapesium	Rusuk sejajar 1 : 8 Rusuk sejajar 2 : 6 Tinggi : 10 Sisi kiri : 6 Sisi kanan : 8	Keliling : sisi kiri + sisi kanan + (jumlah rusuk sejajar) $\rightarrow 6 + 8 + 8 + 6 = 28$ Luas : $0.5 \times (\text{jumlah rusuk sejajar}) \times \text{tinggi}$ $\rightarrow 0.5 \times (8 + 6) \times 10$ $\rightarrow 7 \times 10 = 70$
5.	Jajar Genjang	Alas : 5 Tinggi : 3 Sisi miring : 4	Keliling : $2 \times (s + a)$ $\rightarrow 2 \times (4 + 5)$ $\rightarrow 2 \times 9 = 18$ Luas : $a \times t$

			$\rightarrow 5 \times 3 = 15$
6.	Belah Ketupat	Diagonal 1 : 2 Diagonal 2 : 4 Sisi : 5	Keliling : $4 \times s$ $\rightarrow 4 \times 5 = 20$ Luas : $0.5 \times (d1 \times d2)$ $\rightarrow 0.5 \times (2 \times 4)$ $\rightarrow 0.5 \times 8 = 4$
7.	Layang-Layang	Diagonal 1 : 8 Diagonal 2 : 6 Sisi miring kiri : 10 Sisi miring kanan : 12	Keliling : $2 \times (s + s)$ $\rightarrow 2 \times (10 + 12)$ $\rightarrow 2 \times 22 = 44$ Luas : $0.5 \times (d1 \times d2)$ $\rightarrow 0.5 \times (8 \times 6)$ $\rightarrow 0.5 \times 48 = 24$
8.	Lingkaran	Jari - jari : 7	Keliling : $2 \times 3.14 \times \text{jari-jari}$ $\rightarrow 2 \times 3.14 \times 7$ $\rightarrow 6.28 \times 7 = 43.96$ Luas : $3.14 \times \text{jari-jari} \times \text{jari-jari}$ $\rightarrow 3.14 \times 7 \times 7$ $\rightarrow 21.98 \times 7 = 153,86$

Tabel 15 Data Simulasi Bangun Datar

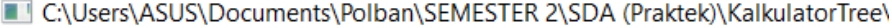
BAB 5

HASIL AKHIR PROGRAM

A. EKSPERIMEN / PENGUJIAN

1. Pengujian Kalkulator

- Inputan user : 2^4-5



```
Aplikasi Kalkulator

Lakukan perhitungan :

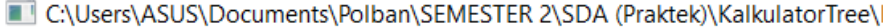
2^4-5

= 11.00

Press any key to continue . . .
```

Gambar 2 Hasil Pengujian 1

- Inputan user : $(0.53+9.10)-2^3$



```
Aplikasi Kalkulator

Lakukan perhitungan :

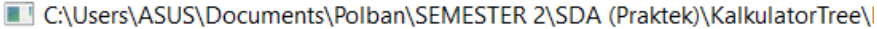
(0.53+9.10)-2^3

= 1.63

Press any key to continue . . .
```

Gambar 3 Hasil Pengujian 2

- Inputan user : $9:3+(1*(4^2))-1$



```
Aplikasi Kalkulator

Lakukan perhitungan :

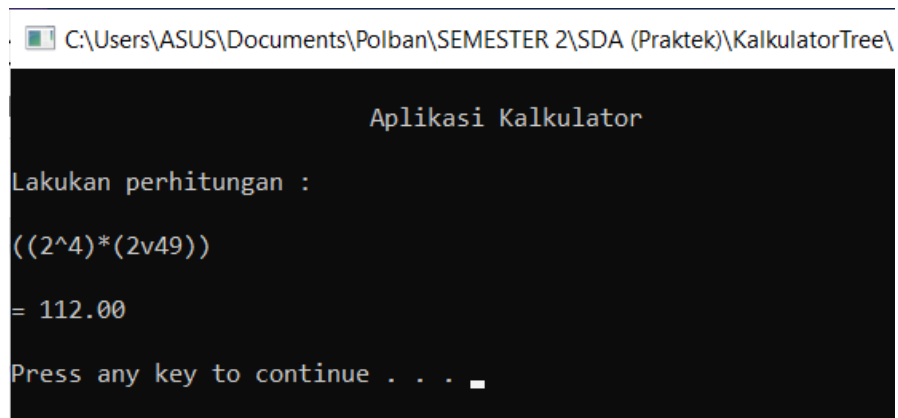
9:3+(1*(4^2))-1

= 18.00

Press any key to continue . . .
```

Gambar 4 Hasil Pengujian 3

- Inputan user : $((2^4)*(2v49))$



```

C:\Users\ASUS\Documents\Polban\SEMESTER 2\SDA (Praktek)\KalkulatorTree\

Aplikasi Kalkulator

Lakukan perhitungan :

((2^4)*(2v49))

= 112.00

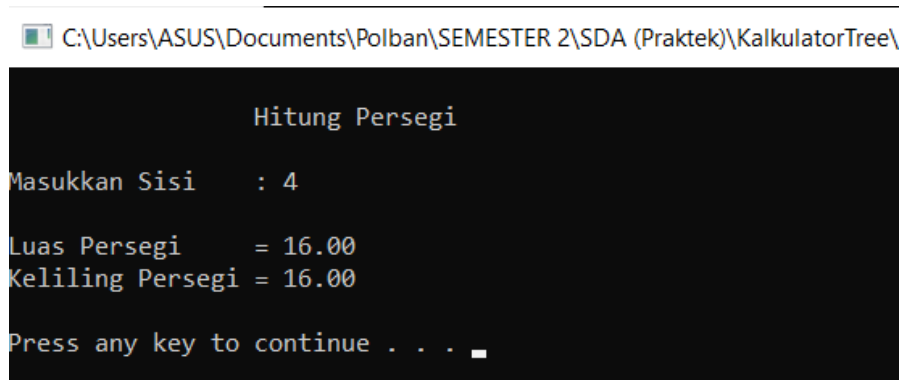
Press any key to continue . . .

```

Gambar 5 Hasil Pengujian 4

2. Pengujian Bangun Datar

- Persegi
Sisi : 4



```

C:\Users\ASUS\Documents\Polban\SEMESTER 2\SDA (Praktek)\KalkulatorTree\

Hitung Persegi

Masukkan Sisi : 4

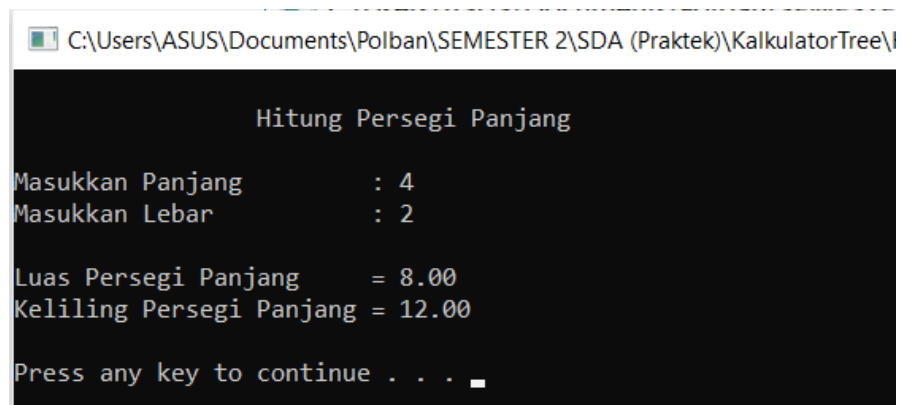
Luas Persegi = 16.00
Keliling Persegi = 16.00

Press any key to continue . . .

```

Gambar 6 Hasil Pengujian Persegi

- Persegi Panjang
Panjang : 4, Lebar : 2



```

C:\Users\ASUS\Documents\Polban\SEMESTER 2\SDA (Praktek)\KalkulatorTree\

Hitung Persegi Panjang

Masukkan Panjang : 4
Masukkan Lebar : 2

Luas Persegi Panjang = 8.00
Keliling Persegi Panjang = 12.00

Press any key to continue . . .

```

Gambar 7 Hasil Pengujian Persegi Panjang

- Segitiga
Alas : 6, Tinggi : 5

```

Hitung Segitiga

Masukkan Alas          : 6
Masukkan Tinggi        : 5

Luas Segitiga           = 15.00
Keliling Segitiga       = 18.00

Press any key to continue . . .

```

Gambar 8 Hasil Pengujian Segitiga

- Trapesium
Rusuk sejajar 1 : 8, Rusuk sejajar 2 : 6, Tinggi : 10, Sisi kiri : 6, Sisi kanan : 8

```

Hitung Trapesium

Masukkan Rusuk Sejajar 1 : 8
Masukkan Rusuk Sejajar 2 : 6
Masukkan Tinggi          : 10
Masukkan Sisi Kiri        : 6
Masukkan Sisi Kanan       : 8

Luas Trapesium            = 70.00
Keliling Trapesium        = 28.00

Press any key to continue . . .

```

Gambar 9 Hasil Pengujian Trapesium

- Jajar Genjang
Alas : 5, Tinggi : 3, Sisi miring : 4

```

Hitung Jajar Genjang

Masukkan Alas          : 5
Masukkan Tinggi        : 3
Masukkan Sisi Miring    : 4

Luas Jajar Genjang      = 15.00
Keliling                = 18.00

Press any key to continue . . .

```

Gambar 10 Hasil Pengujian Jajar Genjang

- Belah Ketupat
Diagonal 1 : 2, Diagonal 2 : 4, Sisi : 5

```

Hitung Belah Ketupat

Masukkan Diagonal 1      : 2
Masukkan Diagonal 2      : 4
Masukkan Sisi             : 5

Luas Belah Ketupat       = 4.00
Keliling Belah ketupat   = 20.00

Press any key to continue . . .

```

Gambar 11 Hasil Pengujian Belah Ketupat

- Layang - Layang
Diagonal 1 : 8, Diagonal 2 : 6, Sisi miring kiri : 10, Sisi miring kanan : 12

```

Hitung Layang-Layang

Masukkan Diagonal 1      : 8
Masukkan Diagonal 2      : 6
Masukkan Sisi Miring Kiri : 10
Masukkan Sisi Miring Kanan : 12

Luas Layang-layang       = 24.00
Keliling Layang-layang   = 44.00

Press any key to continue . . .

```

Gambar 12 Hasil Pengujian Layang-Layang

- Lingkaran
Jari - jari : 7

```

Hitung Lingkaran #

Masukkan Jari-Jari       : 7

Luas Lingkaran           = 153.86
Keliling Lingkaran       = 43.96

Press any key to continue . . .

```

Gambar 13 Hasil Pengujian Lingkaran

B. LESSON LEARNED

1. Fiora Berliana Putri - 201524045
Dari tugas besar ini saya belajar mengenai bagaimana cara kalkulator dapat bekerja, saya juga semakin mengerti penggunaan ADT Tree dan ADT Stack.
2. Lamda Richo Vanjaya Sumaryadi - 201524049
Semakin memahami konsep tree dan stack dalam implementasinya dan semakin terlatih untuk teliti dalam membuat code yang kompleks.
3. Muhamad Aryadipura Sasmita Atmadja – 201524054
Tugas besar ini mengajarkan saya bagaimana pentingnya peran setiap ADT yang ada pada suatu program terutama program kalkulator yang kami buat. Pemahaman saya terhadap tree dan stack juga menjadi semakin matang. Selain itu saya juga menjadi lebih lancar dalam menggunakan GitHub.

DAFTAR PUSTAKA

- [1] C. M. Anjasmara, "Materi Prefix, infix, dan postfix," Blogger, 28 October 2015. [Online]. Available: <http://chandra-mahardika.blogspot.com/2015/10/materi-prefix-infix-dan-postfix.html>. [Accessed 17 July 2021].
- [2] Cendana, "INFIX, PREFIX, POSTFIX," Blogger, 1 October 2015. [Online]. Available: <http://cendana25.blogspot.com/2015/10/infix-prefix-postfix.html>. [Accessed 17 July 2021].
- [3] ChACiooh-GitHub, "ChACiooh Binary Tree Calculator," GitHub, 30 April 2015. [Online]. Available: <https://github.com/ChACiooh/Binary-Tree-Calculator>. [Accessed 19 July 2021].
- [4] Y. P. De, "Infix to Postfix Conversion The Easy Way," Youtube, 12 February 2015. [Online]. Available: <https://youtu.be/vXPL6UavUeA>. [Accessed 22 July 2021].
- [5] D. R. C. K, "PROJECT UAS: Mengubah Notasi infix menjadi notasi prefix dan postfix & Menggunakan algoritma stack," Youtube, 16 June 2021. [Online]. Available: <https://youtu.be/L8MrsLLLve8>. [Accessed 22 July 2021].
- [6] F. Ridwan, "Pengertian dan Contoh Pseudocode Algoritma Tree dalam Struktur Data," Isallab.com, 3 November 2019. [Online]. Available: <https://isallab.com/article/pengertian-dan-contoh-pseudocode-algoritma-tree-struktur-data-part1/#4>. [Accessed 22 July 2021].
- [7] Risa, "Notasi Prefix, Infix, dan Postfix," Blogger, 26 February 2013. [Online]. Available: <http://risasisteminformasi.blogspot.com/2013/02/notasi-prefixinfix-dan-postfix.html>. [Accessed 17 July 2021].
- [8] A. Thakur, "Convert Infix to Postfix Expression," Tutorialspoint, 11 July 2018. [Online]. Available: <https://www.tutorialspoint.com/Convert-Infix-to-Postfix-Expression>. [Accessed 22 July 2021].