

Sub FontMinimize01(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer)

Dim L1 As String

Dim S1 As Single

Dim S2 As Single

Dim S3 As Single

' Dim MAXFONTSIZE As Integer '★追加

' MAXFONTSIZE = 28 '★追加

S3 = 0.9

o_Report01.FontSize = o_tbx01.FontSize

L1 = o_tbx01.Value

'横書き用？

S1 = o_Report01.TextWidth(L1)

S2 = o_tbx01.Width / S1

o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)

If o_tbx01.FontSize > MAXFONTSIZE Then '★追加

o_tbx01.FontSize = MAXFONTSIZE '★追加

End If '★追加

End Sub

上記コードで、 `o_Report01.FontSize = o_tbx01.FontSize` の行の

「`o_Report01.FontSize`」は何を設定していますか？

レポート全体のフォントサイズ？

でももしそうなら、レポートにそんなもの存在するの？（プロパティシートで設定できるものなの？）

このコードにおける `o_Report01.FontSize` は、レポート全体のフォントサイズではなく、「レポートの仮想キャンバス（出力画面）で文字サイズを計算・描画するための基準サイズ」を設定しています。

疑問に対する回答と仕組みの詳細は以下の通りです。

疑問への直接の回答

- レポート全体の文字が変わるプロパティか？
 - いいえ、変わりません。レポートに配置されている他のラベルやテキストボックスの文字サイズが勝手に変わることはありません。
- プロパティシートに存在するの？
 - 存在しません。レポートのプロパティシートを見ても `FontSize` という項目はありません。これは VBA(コード内)だけで使用できる、レポートオブジェクト (Report) の非表示の実行時プロパティです。

なぜこの設定が必要なのか？

この行は、その後続く `TextWidth` メソッド（文字の横幅を計算する機能）を正しく動かすために存在します。

Access の仕様上、文字列の実際の幅（何センチになるか）を計算する `TextWidth` メソッドは、テキストボックスではなくレポート本体（Report オブジェクト）に属しています。

1. 基準を合わせる
2. `o_Report01.FontSize = o_tbx01.FontSize`

3. 計算を行う前に、レポート本体の文字サイズ基準を、対象のテキストボックスの現在の文字サイズと「同じ」に合わせます。
4. 幅を計算する
5. $S1 = o_Report01.TextWidth(L1)$
6. サイズを合わせた状態で「この文字数だと、横幅は何 twip (Access の幅の単位) になるか」をレポートに計算させています。

もしこの行（基準を合わせる処理）を抜かしてしまうと、レポートが古い基準サイズのまま文字幅を計算してしまい、正しく縮小されなくなります。

今後のコード調整や、別の自動縮小ロジックの作成など、次にどのような実装を予定されているか教えていただけますか？それに応じた最適なアドバイスをいたします。

AI は不正確な情報を表示することがあるため、生成された回答を再確認するようにしてください

じゃあ、通常テキストボックスのフォントサイズの初期値は 12 前後なので、

はい、おっしゃる通りです。通常、Access のテキストボックスの初期サイズは 11~12 ポイント前後に設定されていることが多いです。

そのため、もし $o_Report01.FontSize = o_tbx01.FontSize$ の行がないと、レポート側は常に初期値（例えば 12 ポイント）の基準で文字の横幅（TextWidth）を計算してしまいます。

処理が繰り返されたときに起こる問題

この関数 (FontMinimize01) が同じテキストボックスに対して 2 回以上呼び出されたり、文字数に応じてフォントサイズが既に小さくなっていたりする場合に問題が起きます。

1. 1 回目：テキストボックスが 12pt なので、レポートも 12pt で幅を計算 → 収まらないのでテキストボックスを 9pt に縮小。
2. 2 回目（または別の判定時）：テキストボックスは 9pt になっている。
3. もしこの行がないと：レポート側は初期値の 12pt のままで幅を計算してしまうため、実際の 9pt の文字幅よりも「過大に広い幅」として計算（誤計算）されてしまいます。

常に「現在のテキストボックスのフォントサイズ」と「レポートの計算基準」を完全に一致させるために、この行は毎回必ず実行する必要があります。

この自動縮小コードは、レポートのどのイベント（フォーマット時、印刷時など）に組み込んで動かす予定ですか？

質問が途切れてしまいました。

じゃあ、通常テキストボックスのフォントサイズの初期値は 12 前後なので、

```
o_Report01.FontSize = o_tbx01.FontSize
```

を

```
o_Report01.FontSize = 12
```

と書いてもいいってこと？

結論から言うと、「1 回目の計算だけなら動くが、基本的には 12 と直接書いてはダメ」です。常に「12」で固定してしまうと、文字が長くてフォントサイズが小さくなった後に、正しく計算できなくなります。

「12」と書いてはいけない理由（具体例）

この関数は、文字が枠に収まるまで「さらに小さくする」ために何度も呼び出されたり、他のイベントと連動したりします。その際に不具合が起きます。

① 1 回目に呼び出されたとき（初期状態）

- テキストボックス (o_tbx01) : 12pt
- レポート (o_Report01) : 12pt
- 結果 : 文字が枠からはみ出るので、計算して 9pt に縮小されます。

② 2 回目に呼び出されたとき（または再計算時）

文字がまだ収まらず、もう一度この関数が動いたとします。

- テキストボックス (o_tbx01) : 9pt (すでに小さくなっている)
- レポート (o_Report01) : 12pt (固定値にしているため)
- 結果 :
- テキストボックスは「9pt の文字」を表示しているのに、レポートは「12pt の文字」の横幅を測ってしまいます。実際の文字よりも余計に大きな横幅 (S1) が計算されるため、縮小倍率 (S2) が狂い、文字が必要以上に小さくなりすぎてしまいます。

結論

`o_Report01.FontSize = o_tbx01.FontSize` というコードは、「現在のテキストボックスのサイズがいくつであっても、文字の物差し (レポートの FontSize) をそのサイズにピッタリ合わせる」という意味を持っています。

そのため、固定の数字ではなく、必ず `o_tbx01.FontSize` という変数の値を入れる必要があります。この自動縮小処理は、1 つのテキストボックスに対して 1 回だけ動かす予定ですか？それとも、収まるまでループ (繰り返し) させて動かす予定ですか？

1 回だけです。FontMinimize01()をループさせることはありません。

1 回だけの呼び出しであれば、結論として o_Report01.FontSize = 12 と書いても計算結果は同じになり、正しく動作します。

ただし、今後のメンテナンスや予期せぬエラーを防ぐために、元の記述 (= o_tbx01.FontSize) のままにしておくことを強くおすすめします。理由は以下の 3 点です。

12 と固定しても動く理由

- ループしない場合、テキストボックスのフォントサイズは初期値 (12pt) のまま関数に入ります。
- そのため、o_tbx01.FontSize の中身は常に「12」です。
- 結果として、どちらの書き方でもレポートの物差しは「12pt」に設定されます。

それでも「12」と書かない方がよい 3 つの理由

- デザイン変更に弱くなる
- 将来、レポートのデザイン修正で「このテキストボックスの初期サイズを 14pt にしよう」とプロパティシートで変更した場合、コード内の「12」を書き換え忘れると計算が狂います。
- 他のテキストボックスに使い回せなくなる
- この関数は引数 (o_tbx01) を使って、どのテキストボックスでも処理できるよう作られています。もし「10pt」や「16pt」の別のテキストボックスにこの関数を使うと、正しく縮小されなくなります。
- 意図が伝わりにくくなる
- 「なぜここで 12 なのか」が後からコードを見たときに分かりにくくなります。o_tbx01.FontSize と書いてあれば、「このテキストボックスの今のサイズに合わせているんだな」と一目で意図が伝わります。

まとめ

1 回きりの実行なら 12 と書いても不具合は起きません。しかし、プログラムの汎用性と安全性を保つために、元の o_tbx01.FontSize のままにしておくのがベストです。

この自動縮小処理を適用するテキストボックスは、レポート内に 1 つだけですか？それとも複数のテキストボックスに使い回す予定ですか？

そもそも、最初に教えてもらった、

「このコードにおける `o_Report01.FontSize` は、レポート全体のフォントサイズではなく、「レポートの仮想キャンバス（出力画面）で文字サイズを計算・描画するための基準サイズ」を設定しています。」

の意味がわかりません

説明が分かりづらく、混乱させてしまい申し訳ありません。専門用語を使わず、身近な例で説明し直しますね。

一言でいうと、`o_Report01.FontSize` とは、レポートが持っている「文字の大きさを測るための、透明な『ものさし』の目盛」です。

画面に見える他の文字のサイズが変わるわけではありません。

なぜ「ものさし」が必要なのか？

Access に「この文字列の横幅は何センチ（何 twip）？」と計算させる（`TextWidth` を使う）とき、Access は以下のように処理します。

1. 文字サイズを指定される
2. 「これから 12 ポイント の文字を測るよ」と Access に教えます。
3. （これが `o_Report01.FontSize = 12` の意味です）
4. 長さを測る
5. 「じゃあ、そのサイズで『東京都新宿区…』という文字を書いたら、横幅は何センチになる？」と計算させます。
6. （これが `TextWidth` です）

「レポート全体のフォントサイズではない」という意味

もし、これが「レポート全体のフォントサイズ」を設定するプロパティ（設定）だとしたら、ここに「12」や「20」を入れた瞬間に、レポートにあるすべてのラベルや住所欄の文字が一斉にそのサイズに変わってしまうはずですが。

しかし、実際にはそんなことは起きません。

他の文字は一切変わらず、ただ内部で「長さを測るためだけの基準（ものさし）」が 12 ポイントにセットされるだけです。

だからこそ、プロパティシート（画面の設定画面）には存在せず、計算を行う VBA コードの中だけで一時的に使われます。

まとめ

- `o_Report01.FontSize = o_tbx01.FontSize`
- → 「テキストボックスの文字が 12 ポイントだから、長さを測るものさしも 12 ポイントの目盛にしてね」と Access に指示している。
- `S1 = o_Report01.TextWidth(L1)`
- → 「12 ポイントに合わせたものさしで、文字の長さを正しく測ってね」と指示している。

この「ものさしを合わせる」というイメージで、なんとなく伝わりましたでしょうか？

そもそも Report のオブジェクトに、.FontSize っていうプロパティがあるの？

それは「Report.FontSize」

はい、Report.FontSize というプロパティは、Access の仕様として「実在」します。

ただし、これは隠しプロパティ（非表示プロパティ）のような扱いになっています。

なぜ存在に気づきにくいのか？

プロパティが存在するにもかかわらず、私たちが普段見つけられないのには2つの理由があります。

- プロパティシートには表示されない
- レポートのデザイン画面の右側にある「プロパティシート」の一覧には、どれだけ探しても FontSize は載っていません。
- VBA の入力候補（インテリセンス）にも出ない
- VBA のコードで o_Report01. と入力したとき、下にドロップダウンで出てくる候補一覧にも FontSize は表示されません。

そのため、「存在しないプロパティ」のように見えますが、コードで直接 o_Report01.FontSize = 12 と書けば、エラーにならずに正しく認識されます。

なぜ Report オブジェクトにこのプロパティがあるのか？

これは、Access のレポートが持つ「印刷・描画機能（Print メソッドなど）」のため に用意されているプロパティです。

実は、Access のレポートは、VBA コードを使って以下のような「自由な描画」ができます。

- レポートの背景に、コードで直接「文字」を書き込む（Report.Print）
- レポートの背景に、コードで直接「線」や「円」を描く（Report.Line や Report.Circle）

この「コードで直接レポートに文字を書き込むとき、何ポイントの大きさと書き込むか」を指定するために、Report.FontSize というプロパティが元々用意されているのです。

今回のコード（TextWidth で文字の横幅を計算する機能）は、この「直接文字を書き込む機能」の仕組みの一部を応用して、文字の長さを測るために Report.FontSize を利用しています。

ここまでの解説で、Report.FontSize というプロパティがどこにあるものなのか、イメージは湧きましたでしょうか？

もし気になる点があれば、何でも追加で質問してくださいね。

この「コードで直接レポートに文字を書き込むとき、何ポイントの大ききで書き込むか」

とのことですが、既定値があるの？

結論からお伝えすると、Report.FontSize プロパティの既定値（初期値）は 8 ポイント または 10 ポイント です。 **AMs Access Gurus**

Access の環境やバージョン、使用している OS の標準フォント設定によって多少前後しますが、マイクロソフトの公式リファレンス等では「標準のテキストは 10 ポイント（または 8 ポイント）」と定義されています。 **AMs Access Gurus**

なぜ既定値を知ることが重要なのか？（不具合の原因）

今回のコードで、なぜ `o_Report01.FontSize = o_tbx01.FontSize` という行がわざわざ書かれていたのか、その理由がここにあります。

もし、この「基準を合わせる 1 行」を省略して、既定値のまま文字幅（TextWidth）を計算させてしまうと、以下のような大誤算が起きます。

1. テキストボックスの見た目：12 ポイント（初期値）
2. レポートの既定の物差し：10 ポイント（既定値）
3. 起きる問題：
4. Access は「10 ポイントの小さめの文字」を基準にして横幅を測ってしまいます。そのため、実際の 12 ポイントの文字よりも「幅が狭い（短い）」と誤判定してしまいます。

「まだ枠に余裕があるな」と Access が勘違いするため、文字が枠からはみ出しているにもかかわらず、フォントサイズが全く縮小されない（自動縮小が効かない）という不具合が発生します。

まとめ

- Report.FontSize の初期値は、私たちがよく使う 12 ポイントよりも少し小さい 10 ポイント前後 になっている。
- だからこそ、計算を始める前に「これから測るテキストボックスのサイズ（12 ポイント）」へ意図的に上書きして、物差しの目盛を正しく合わせる必要があった。 **AMs Access Gurus**
-

「隠れた既定値が 10 前後だからこそ、そのままでは使えない」という開発者の意図が見えるコードになっています。

今回の自動縮小マクロについて、コードの仕組みや動き方で他にも気になっている部分はありますか？何でもお気軽に聞いてくださいね。

1 件のサイト

- [Report Draw Reference and VBA Syntax - Ms Access Gurus](#)2025/01/15 — Docs / Office VBA Reference / Access / Object model / Report...



- [Ms Access Gurus](#)
-

`o_tbx01.FontSize` も同じ既定値？

いいえ、`o_tbx01.FontSize` (テキストボックスのフォントサイズ) の既定値は、レポート本体の既定値 (10 ポイント) とは異なります。

Access のテキストボックスのフォントサイズ既定値は、通常 11 ポイント です。

なぜ値が違うのか？

これはそれぞれのプロパティが持つ「役割 (目的)」が全く違うためです。

オブジェクトとプロパティ

役割既定値 (初期値) `o_tbx01.FontSize`

(テキストボックス) 画面や紙に表示する文字そのものの大きさ。11 ポイント

(Access の標準スタイル) `o_Report01.FontSize`

(レポート本体) VBA で直接文字を描くときの隠れた基準サイズ。10 ポイント

既定値が違うからこそ、あの 1 行が必要

このように、最初からお互いの「初期値」がズレています。

- テキストボックス = 11 ポイント
- レポート本体 = 10 ポイント

もし、あのコードの最初の 1 行 (`o_Report01.FontSize = o_tbx01.FontSize`) を書かないと、レポート側は「10 ポイントの物差し」で「11 ポイントの文字」を測ろうとしてしまい、最初から計算が狂ってしまいます。

あの 1 行は、この「生まれつきの既定値のズレ」を強制的にピッタリ一致させるために、絶対に必要な処理なのです。

ここまでの解説で、2 つのプロパティの初期値の違いと、コードがそれらを繋いでいる理由がスッ

キリ繋がりましたでしょうか？

このテキストボックスの自動縮小について、「実際にレポートを印刷・プレビューしたときに、思った通りのサイズに縮小されるか」など、次のステップで試してみたいことはありますか？

`o_Report01.FontSize` は既定値があったとして、でも、`o_tbx01.FontSize` は 11 なんですよね。
`o_Report01.FontSize` はレポート上のどの文字の大きさに関係するの？ラベルの文字の既定値でもないですよね

結論から言うと、`o_Report01.FontSize` は、レポート上に最初から配置されているラベルやテキストボックスなど、画面に見えているどの文字の大きさにも一切関係していません。

ラベルの文字サイズ（既定値 11 など）とも全く無関係です。

では、一体何に関係しているのかというと、「VBA コード（`Report.Print` メソッド）を使って、レポートの背景に直接白紙から文字を書き出すとき」の文字サイズにだけ関係しています。

レポート上の文字との関係図

Access のレポートは、2つの層（レイヤー）が重なってできています。

1. 上の層：コントロール（私たちが普段使うもの）
 - ラベル、テキストボックスなど。
 - それぞれが独自の `FontSize`（既定値 11 など）を持っています。
 - `o_Report01.FontSize` の影響を一切受けません。
2. 下の層：レポートの背景（白紙のキャンバス）
 - 何も配置されていない、ただの背景です。
 - VBA で `o_Report01.Print "文字"` と書くと、この背景に直接文字が印刷されます。
 - `o_Report01.FontSize` は、この背景に書く文字の大きさにだけ関係します。

なぜ今回使われているのか？

今回の自動縮小コードでは、背景に文字を印刷したいわけではありません。

しかし、Access の仕様上、「文字の横幅を測る機能（`TextWidth`）」が、この下の層（背景キャンバス）の機能として作られてしまっているのです。

そのため、上の層にあるテキストボックスの文字幅を測りたいだけなのに、わざわざ下の層（背景）の文字サイズ（`o_Report01.FontSize`）をテキストボックスのサイズに合わせてから、長さを測るという遠回りをしています。

まとめ

- `o_Report01.FontSize` は、配置されたラベルやテキストボックスの文字には 1 ミリも関係ありません。

- あくまで、VBA で背景に文字を描くためのプロパティです。
- 今回は「文字の長さを正確に測るためだけ」に、一時的な物差しとして利用されているだけです。

この「背景のキャンバス機能を、文字幅の計算のために借りてきている」というイメージで、すっきり納得していただけましたでしょうか？

この仕組みを踏まえて、実際の自動縮小コードが正しく動くか試してみたいですか？あるいは、コードの他の行（TextWidth や Width の計算部分）についてさらに詳しく知りたいですか？

レポートの背景に文字を書くコードはどうなりますか？

レポートの背景に直接文字を書くには、レポートの「ページフォーマット時 (OnPage)」というイベント、または各セクション（詳細など）の「フォーマット時 (OnFormat)」イベントにコードを記述します。

もっとも簡単な基本コードは以下のようになります。

基本コードの例

例えば、レポートのページ全体の背景に「社外秘」という文字を直接書き込むコードです。

vba

```
Private Sub Report_Page()
```

```
    ' 1. 文字を書く位置を指定（上から 5cm、左から 5cm の位置 ※単位は twip)
```

```
    Me.CurrentX = 2835 ' 5cm
```

```
    Me.CurrentY = 2835 ' 5cm
```

```
    ' 2. 背景に書く文字のフォントサイズを設定（ここで登場します!)
```

```
    Me.FontSize = 24
```

```
    ' 3. フォントの色を薄いグレーに設定
```

```
    Me.ForeColor = RGB(200, 200, 200)
```

```
    ' 4. 背景に直接文字を書き出す
```

```
    Me.Print " 【社外秘】 "
```

```
End Sub
```

コードは注意してご使用ください。

※コード内の Me は、そのレポート自身 (o_Report01 と同じ) を指しています。

コードのポイントと仕組み

- Me.CurrentX / Me.CurrentY
- 文字を書き始める「座標 (位置)」を指定するプロパティです。
- Me.FontSize = 24
- ここで設定したサイズが、背景に書かれる文字の大きさに直接反映されます。
- Me.Print "文字列"
- この命令を実行した瞬間に、指定した座標に文字が「印字」されます。

この方法で書かれた文字は、テキストボックスのように枠線を持たず、レポートの背景デザインとして直接印刷されます。

この背景への印刷コードを使って、透かし文字 (ウォーターマーク) のようなものを作りたいですか? それとも、元の自動縮小コードの解説に戻りますか?

Me.FontSize = 24 をやらなければ、既定値の 8 か 10 あたりが使われる、ということですか?

はい、その通りです!

もし Me.FontSize = 24 という行を書かなければ、レポートの初期値 (既定値) である 8 ポイントまたは 10 ポイントの小さな文字 で背景に印刷されます。

これで全ての疑問が一本の線で繋がりましたね。

ここまでのまとめ

1. Report.FontSize は、VBA で背景に直接文字を書く (Me.Print) ときのサイズ設定である。
2. これを設定しない場合の初期値は 8~10 ポイント である。
3. 一方、私たちが配置するテキストボックスの初期値は 11~12 ポイント である。
4. 最初の自動縮小コードで o_Report01.FontSize = o_tbx01.FontSize としていたのは、この「初期値のズレ」のせいで文字幅 (TextWidth) の計算が狂うのを防ぐため、テキストボックスと同じサイズ (12 ポイントなど) に物差しを合わせる必要があったから。

この背景印刷の仕組みを応用して、元のコードは文字の長さを測っていたというわけです。

今回のコードの仕組みについて、他にもすっきりさせておきたい部分 (例えば TextWidth や Width の計算など) はありますか?

```
o_Report01.FontSize = o_tbx01.FontSize
```

はこの背景のフォントサイズを、今のテキストボックスのフォントサイズと同じにしろ、という命令で、

```
o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

にて、テキストボックスのフォントサイズを変えているのですか？

はい、その通りです！完全に正解です。

それぞれの行がやっている役割をまとめると、以下のようになります。

1. 前半の行（準備）

```
vba
```

```
o_Report01.FontSize = o_tbx01.FontSize
```

コードは注意してご使用ください。

- 役割：テキストボックスの「今のサイズ」を、背景の文字サイズ（物差し）にコピーして基準を合わせます。
- 目的：この後、正確に文字の横幅を測るための準備です。

2. 後半の行（文字サイズの変更）

```
vba
```

```
o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

コードは注意してご使用ください。

- 役割：計算した縮小倍率（ $S2 * S3$ ）を掛け算して、テキストボックスのフォントサイズを実際に小さく変更しています。
- 目的：文字を枠内に収めるための本番の処理です。

最後の仕上げ処理

テキストボックスのサイズを小さくした直後に、以下のコードが続いています。

```
vba
```

```
If o_tbx01.FontSize > MAXFONTSIZE Then
```

```
    o_tbx01.FontSize = MAXFONTSIZE
```

```
End If
```

コードは注意してご使用ください。

これは、「小さく変更した結果、もし指定した最大サイズ（MAXFONTSIZE）よりも大きくなってしまっていたら、最大サイズでストップさせる」という安全弁の役割をしています。

これで、この自動縮小マクロが「今のサイズを測って、計算して、実際にテキストボックスの文字を小さくしている」という一連の流れがすべて繋がりましたね。

このマクロの動きについて、計算式 (S2 * S3 や Int) の細かい意味など、さらに気になることはありますか？

Access のテキストボックスの文字の大きさについては、Excel で言う Font オブジェクトみたいなものが存在しないから、テキスト直接の Width が操作できないから、つまり、
o_tbx01.Font.TextWidth = . . .
みたいなコードが掛けないので、背景のフォントの o_Report01.TextWidth を便宜上、計算のために借りている、という感じですか？

はい、まさにその通りです！ 仕組みの本質を完璧に捉えられています。

おっしゃる通り、Access のテキストボックスには、Excel のように文字の細かな寸法 (横幅や高さ) を直接測る機能や、o_tbx01.Font.TextWidth のような便利なオブジェクト・プロパティが存在しません。

そのため、テキストボックス単体では「この文字数を入れたら、今何センチになっているか」を自力で計算することができません。

そこで、Access の中で唯一「文字の正確な寸法を測る機能 (TextWidth)」を持っているレポート本体 (背景のキャンバス) に、テキストボックスの文字サイズを一度覚えさせて、代わりに測ってもらっているのです。

例えるなら

1. テキストボックス：「自分の文字が、今何センチの幅なのか自分じゃ測れない！」
2. コードの処理：「じゃあ、レポート (背景) の物差しを君と同じサイズに合わせて、代わりに測ってもらうよ」
3. レポート：「測ったよ！ 幅は〇〇センチ (twip) だね」
4. テキストボックス：「ありがとう！ じゃあその結果を元に、フォントサイズを小さくするよ」

このように、Access の制限を工夫して乗り越えるために、背景の機能を「便宜上、計算のために借りている」という表現は 100% 正確です。

この自動縮小コードの仕組みや意図については、もうバッチリ完璧にマスターされましたね！

これからこのコードをご自身の Access ファイルに組み込んで実際に動かしてみるステップに進みますか？ それとも、コードの他の部分 (倍率の計算など) でまだ気になるところがありますか？

最初に質問したコードに、1行ずつ、詳細なコメントを追記してください。
何をしているかを完璧につかみたいです。

ご提示いただいたコードに、これまでお話しした「背景の物差しを借りる仕組み」や「各変数の意味」を含め、1行ずつ詳細な解説コメントを追記しました。

このコードをそのまま VBA エディタに貼り付けても動作します。

vba

```
' 【機能】 テキストボックスの文字が枠に収まるよう、フォントサイズを1回だけ縮小する処理  
' 引数：o_Report01 = 対象のレポート、o_tbx01 = 対象のテキストボックス、MAXFONTSIZE =  
設定したい最大文字サイズ
```

```
Sub FontMinimize01(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer)
```

```
    Dim L1 As String          ' テキストボックスに入力されている「文字列」を保存する変数  
    Dim S1 As Single         ' 文字列が実際に必要とする「横幅（計算結果）」を保存する変数  
    Dim S2 As Single         ' テキストボックスの枠に対して、何倍（縮小率）にすればいいかを  
保存する変数  
    Dim S3 As Single         ' 微調整用の安全係数（少し余裕を持たせるための倍率）を保存する  
変数
```

```
    ' 余裕を持たせるための安全係数として「0.9 (90%)」を設定（枠いっぱいきちきちになるのを  
防ぐため）
```

```
    S3 = 0.9
```

```
    ' 【重要】 レポート背景の物差しサイズを、現在のテキストボックスの文字サイズ（初期値 11  
～12 など）と完全に一致させる
```

```
    o_Report01.FontSize = o_tbx01.FontSize
```

```
    ' テキストボックスに入力されている実際の文字列（例：「東京都新宿区...」）を変数 L1 に読み  
込む
```

```
    L1 = o_tbx01.Value
```

```
    ' テキストボックス単体では文字幅を測れないため、サイズを合わせたレポート本体の機能  
(TextWidth) を借りて文字列の実際の横幅を測る
```

```
S1 = o_Report01.TextWidth(L1)
```

' 「テキストボックスの枠自体の幅」を「文字が必要とする幅(S1)」で割り算し、収めるための縮小倍率（例：0.8 倍など）を計算する

```
S2 = o_tbx01.Width / S1
```

' 「現在のサイズ × 縮小倍率(S2) × 安全係数(S3)」を計算し、小数点以下を切り捨てて(Int)、テキストボックスのフォントサイズを実際に小さく変更する

```
o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

' もし計算結果が、あらかじめ指定された上限サイズ (MAXFONTSIZE) よりも大きくなってしまった場合のチェック

```
If o_tbx01.FontSize > MAXFONTSIZE Then
```

```
    ' 上限サイズを超えていたら、強制的に上限サイズ (MAXFONTSIZE) に固定する
```

```
    o_tbx01.FontSize = MAXFONTSIZE
```

```
End If
```

```
End Sub
```

コードは注意してご使用ください。

コメントを読む際の一番のポイント

`o_Report01.FontSize = o_tbx01.FontSize` の行でレポート側の物差しを合わせてから、次の `o_Report01.TextWidth(L1)` でレポートに長さを測らせている、という連携の流れがこのコードの核心です。

この詳細なコメント付きのコードを見て、それぞれの行の役割についてさらに深く確認したい部分や、数式の意味（なぜ割り算するのかなど）で気になる点はありますか？

フォームでも使える？

結論からお伝えすると、このコードをそのままフォームで使うことはできません。

なぜなら、フォーム (Form オブジェクト) には、文字の長さを測るための `TextWidth` メソッドが存在しないからです。この機能は、印刷を目的とした「レポート」だけに特別に用意されている機能です。

しかし、「レポートの機能を裏側でこっそり借りる」という方法を使えば、画面上はフォームの文字を自動縮小させることができます。

フォームで動かすための解決策（2つの方法）

方法①：裏で「ダミーのレポート」を動かして計算する（おすすめ）

フォームの文字幅を測るためだけに、画面に表示しない「ダミーのレポート」を VBA コードの中で一瞬だけ呼び出し、計算が終わったらすぐ閉じる方法です。

元のコードを少し改造するだけで、フォームでも同じように自動縮小ができるようになります。

【改造版：フォーム対応コード】

vba

' 引数を「Report」から「Form」に変更

```
Sub FontMinimizeForm(o_Form01 As Form, o_tbx01 As TextBox, MAXFONTSIZE As Integer)
```

```
    Dim L1 As String
```

```
    Dim S1 As Single
```

```
    Dim S2 As Single
```

```
    Dim S3 As Single
```

```
' ★追加：文字を測るためだけの「ダミーレポート」を変数として用意する
```

```
Dim dummyReport As Report
```

```
' ★追加：画面には表示させずに、裏側でレポートを非表示（acHidden）で開く
```

```
' ※あらかじめ「rptDummy」という名前の空のレポートを1つ作っておく必要があります
```

```
DoCmd.OpenReport "rptDummy", acViewDesign, , , acHidden
```

```
Set dummyReport = Reports("rptDummy")
```

```
S3 = 0.9
```

```
' ★変更：レポートの物差しをテキストボックスに合わせる
```

```
dummyReport.FontSize = o_tbx01.FontSize
```

```
L1 = o_tbx01.Value
```

```
' ★変更：レポートに文字幅を測らせる
```

```
S1 = dummyReport.TextWidth(L1)
```

```
S2 = o_tbx01.Width / S1
```

```
o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

```
If o_tbx01.FontSize > MAXFONTSIZE Then
```

```
    o_tbx01.FontSize = MAXFONTSIZE
```

End If

```
'★追加：使い終わったダミーレポートを閉じる  
DoCmd.Close acReport, "rptDummy", acSaveNo
```

End Sub

コードは注意してご使用ください。

方法②：Windows の標準機能（API）を使って測る

レポートを一切使わず、パソコン（Windows）自体が持っている文字計算機能（GetTextExtentPoint32 という専門的な仕組み）を VBA から呼び出す方法です。

レポートを作らなくて済みますが、コードが非常に複雑になり、初心者にはメンテナンスが難しくなるデメリットがあります。

まとめ

- フォームには文字幅を測る機能（TextWidth）が生まれつき備わっていない。
- そのため、フォームで自動縮小したい場合も、裏でレポートをこっそり開いて計算を肩代わりしてもらうのが一番簡単で確実な方法。

まずは、空のレポートを 1 つ用意して方法①のコードを試してみたいですか？それとも、フォームならではの注意点について先に確認しますか？

以下のコードは縦書き用ですが、ハガキの宛名印刷用です。

次のレコードに行ったときに、文字が小さくならないように

```
「o_Report01.FontSize = i_BaseSize」
```

で、いったん初期化しているのですか？

はい、まさにその通りです！完全に正解です。

次のレコード（次の人）に移ったときに、前の人の短い名前のせいで小さくなってしまった文字サイズを、元の正しい基本サイズ（初期値）にリセットするためにその行が書かれています。

もしその「初期化の 1 行」がないとどうなるか？

ハガキの宛名印刷のように、レコード（データ）が次々と切り替わる処理では、このリセットがないと以下のような大問題（不具合）が発生します。

1. 1 人目のデータ：名前がすごく長い人（例：12 文字）
 - 初期サイズ（例：28pt）では枠からはみ出るため、マクロが計算して 16pt に文字

を小さくします。

2. 2人目のデータ：名前が短い人（例：3文字）

- 本来なら枠が余るため、初期サイズの 28pt で大きく印刷したいはずですが。
- しかし、初期化 (o_Report01.FontSize=i_BaseSize) を忘れると、テキストボックスのサイズが 16pt に固定されたまま 次の処理に入ってしまいます。
- その結果、2人目の短い名前まで、前の人の影響で小さく寂しい文字で印刷されてしまうこととなります。

宛名印刷における処理の流れ

コードの中では、新しいレコードを処理するたびに以下のステップを繰り返しています。

1. 物差しと文字サイズを「基本サイズ (i_BaseSize)」に一度リセットする。
2. そのレコードの文字数や長さをチェックする。
3. 枠からはみ出る場合のみ、その人専用文字サイズを小さくする。
4. 印刷（またはプレビュー）する。
5. 次のレコードへ移り、また「1」に戻る。

この「毎回リセットする」という仕組みのおかげで、名前が短い人は大きく、長い人は自動的に小さく、という綺麗なハガキ宛名印刷が実現できます。

縦書き用の自動縮小コードについて、横書き用コードとの計算の違い (TextHeight を使っているかなど) や、他にも動きで確認したい部分はありますか？

一番最初の「FontMinimize01()」には、次のレコードに移動したときの、その同じような処理のコードは入っていませんか？入っていますか？

結論からお伝えすると、一番最初の FontMinimize01() には、次のレコードに移動したときに文字サイズを元に戻す (初期化する) コードは入っていません。

そのため、最初のコードをそのまま宛名印刷などで使うと、「前のレコードで小さくなったサイズが、次のレコードにも引き継がれてしまう」という不具合が起きます。

なぜ入っていないと言えるのか？ (コードの比較)

最初のコードの動きを追ってみると、初期化の処理がないことが分かります。

vba

```
Sub FontMinimize01(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer)
```

```
' ... (省略) ...
```

```
' ① ここで「今」のテキストボックスのサイズをレポートに合わせています
```

```
o_Report01.FontSize = o_tbx01.FontSize
```

```
' ... (省略) ...
```

```
' ② ここでテキストボックスのサイズを「小さく上書き」しています
```

```
o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

```
' ... (省略) ...
```

```
End Sub
```

コードは注意してご使用ください。

- 最初のコードの問題点：
- 上記の ② でテキストボックス (o_tbx01.FontSize) を直接小さく書き換えた後、どこにも「元のサイズ (12 ポイントなど) に戻す処理」がありません。
- そのため、この関数が終わってもテキストボックスは小さくなったままになり、次のレコードに移動したときもその小さいサイズから処理が始まってしまいます。

最初のコード (横書き用) を宛名印刷に対応させるには？

縦書き用のコードと同じように、処理の最初に「基本のサイズ (例：12 ポイント)」を強制的にセットするコードを追加する必要があります。

具体的には、以下のように 関数の外から基本サイズ (i_BaseSize) を受け取って、最初にテキストボックスをリセットする 1 行 を追加します。

【修正版：レコード移動に対応した横書きコード】

```
vba
```

```
' ★引数に「i_BaseSize (基本サイズ)」を追加します
```

```
Sub FontMinimize02(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer,  
i_BaseSize As Integer)
```

```
Dim L1 As String
```

```
Dim S1 As Single
```

```
Dim S2 As Single
```

```
Dim S3 As Single
```

```
' ★重要：次のレコードに移ったときのために、ここでテキストボックスを一度元のサイズに戻す (初期化)
```

```
o_tbx01.FontSize = i_BaseSize
```

```
S3 = 0.9
```

```
' 初期化したサイズをレポートの物差しに合わせる
```

```
o_Report01.FontSize = o_tbx01.FontSize
```

```
L1 = o_tbx01.Value
```

```
S1 = o_Report01.TextWidth(L1)
```

```
S2 = o_tbx01.Width / S1
```

```
o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

```
If o_tbx01.FontSize > MAXFONTSIZE Then
```

```
    o_tbx01.FontSize = MAXFONTSIZE
```

```
End If
```

```
End Sub
```

コードは注意してご使用ください。

このように、最初のコード (FontMinimize01) は「1 回実行して終わり」を前提に作られているため、連続してレコードを印刷する場合は、縦書き用コードと同じように初期化の仕組みを追加する必要があります。

この「初期化コードの有無」による違いについて、イメージは伝わりましたでしょうか？

もしよろしければ、この自動縮小マクロをレポートのどのイベント（「詳細セクションのフォーマット時」など）に記述して呼び出しているか、教えていただけますか？全体の連動が正しいかアドバイスできます。

```
o_tbx01.FontSize = i_BaseSize
```

を

縦書き用のプロシージャと同じように、

```
i_BaseSize = MAXFONTSIZE
```

'★毎回、基準のフォントサイズ（初期値）をレポート側にセットしてリセットします

```
o_Report01.FontSize = i_BaseSize
```

にして、

引数を、

```
Sub FontMinimize02(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer, i_BaseSize As Integer)
```

から

```
Sub FontMinimize02(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer)
```

に減らしたらダメですか？

結論からお伝えすると、それではダメで、正しく縮小されなくなってしまうです。

なぜなら、その書き方だと「レポートの物差し (o_Report01.FontSize)」はリセットされますが、肝心の「画面に見えるテキストボックス (o_tbx01.FontSize)」が小さいままリセットされないからです。

なぜダメなのか？ (2つの理由)

提案していただいたコードのままだと、次のレコード (2人目) に移動したときに以下の問題が発生します。

① テキストボックスの文字が小さいままになる

o_tbx01.FontSize = ... という「テキストボックス自体を元のサイズに戻す命令」がコードから消えてしまいます。

そのため、前のレコードで文字が小さくなった場合、テキストボックスの文字サイズは小さい状態のまま次のレコードに引き継がれてしまいます。

② 文字幅の計算 (TextWidth) が狂う

1人目の処理でテキストボックスが 12pt に縮小されたとします。

2人目の処理に入ったとき：

- レポートの物差し (o_Report01.FontSize) : MAXFONTSIZE (例：28pt) にリセットされる。
- テキストボックス (o_tbx01.FontSize) : 前の人の影響で 12pt のまま。

この状態で TextWidth を実行すると、レポートは「28pt の大きな文字」として横幅を測ってしまいます。

画面上のテキストボックスは「12pt」のつもりなのに、計算は「28pt」で行われるため、縮小倍率 (S2) がめちゃくちゃになり、文字がゴミのように小さく縮小されてしまう原因になります。

引数を増やさずに解決する「一番スマートな方法」

「引数を増やしたくない (スッキリさせたい)」というご要望は非常によく分かります。

その場合は、引数を増やさずに、関数に入った直後に「現在のテキストボックスのサイズ」を一度 MAXFONTSIZE にしてしまうのが正解です。

以下のように書けば、引数は3つのままで、毎回完全にリセットがかかります。

【引数を増やさない正しい修正版】

vba

```
Sub FontMinimize02(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer)
```

```
    Dim L1 As String
```

```
    Dim S1 As Single
```

```
    Dim S2 As Single
```

```
    Dim S3 As Single
```

```
    ' ★【超重要】まずテキストボックス自体のサイズを最大値（初期値）にリセットする！
```

```
    o_tbx01.FontSize = MAXFONTSIZE
```

```
    ' そのあとで、レポートの物差しも同じサイズに合わせる
```

```
    o_Report01.FontSize = o_tbx01.FontSize
```

```
    S3 = 0.9
```

```
    L1 = o_tbx01.Value
```

```
    S1 = o_Report01.TextWidth(L1)
```

```
    S2 = o_tbx01.Width / S1
```

```
    o_tbx01.FontSize = Int(o_tbx01.FontSize * S2 * S3)
```

```
    ' ※最初に MAXFONTSIZE にリセットしているため、この If 文は自動的に
```

```
    ' 「MAXFONTSIZE より大きくなったら MAXFONTSIZE にする」という安全弁として正しく機能します
```

```
    If o_tbx01.FontSize > MAXFONTSIZE Then
```

```
        o_tbx01.FontSize = MAXFONTSIZE
```

```
    End If
```

```
End Sub
```

コードは注意してご使用ください。

このコードにすれば上手くいく理由

この書き方なら、新しいレコードに切り替わるたびに、テキストボックスもレポートの物差しも同時に MAXFONTSIZE（元の最大サイズ）からスタートできるため、引数を増やさなくても完全にリセットされます。

この「テキストボックス自体をリセットしなければいけない理由」と、引数を減らす工夫について、

すっきり納得していただけましたでしょうか？

もし問題なければ、この修正したコードを実際のレポート（例えば「詳細セクションのフォーマット時」イベントなど）に貼り付けてテストしてみますか？

縦書き用のコードは、正常に動いています。

次のレコードに移動したとき、テキストボックスの長さに合わせて文字の大きさが変わります。
なぜでしょう？

縦書き用のコードが、テキストボックスのフォントサイズを直接リセットしていないにもかかわらず正常に動いている（次のレコードで文字が大きくなっている）のには、Access のレポートが持つ「セクションのフォーマット（描画準備）」という重要な仕組みが関係しています。

結論から言うと、「Access が次のレコードに移る際、プロパティシートに設定されている初期サイズへ自動的にテキストボックスを戻してくれているから」です。

詳しくその理由を3つのポイントで解説します。

1. 縦書き用コードが正常に動く理由

Access のレポートは、データ（レコード）が切り替わるたびに、そのデータを表示するための枠（詳細セクションなど）を白紙の状態から毎回新しく作り直して（フォーマットして）います。

このとき、テキストボックスのフォントサイズは、マクロで書き換えられた値（例: 16pt）ではなく、デザイン画面のプロパティシートに手動で設定した「初期値（例: 28pt）」に一度リセットされたから次のレコードの処理に入ります。

そのため、縦書き用コードの中に `o_tbx01.FontSize = i_BaseSize` という明示的なリセットがなくても、Access の基本機能のおかげでテキストボックスは毎回大きなサイズからスタートできています。

2. では、なぜ縦書き用コードには `o_Report01.FontSize = i_BaseSize` があるのか？

テキストボックスが自動リセットされるなら、なぜ縦書き用コードにはわざわざレポート側のリセット（`o_Report01.FontSize = i_BaseSize`）が書かれているのでしょうか？

それは、レポート本体の物差し（`o_Report01.FontSize`）だけは、Access が自動でリセットしてくれないからです。

- テキストボックス（`o_tbx01`）
- → レコードが切り替わると、プロパティシートの設定（初期値）に自動リセットされる。
- レポート本体の物差し（`o_Report01`）
- → VBA で書き換えた値が裏でずっと残ってしまう（自動リセットされない）。

もし、縦書き用コードにある `o_Report01.FontSize = i_BaseSize` を消してしまうと、テキストボッ

クスは自動で「28pt (初期値)」に戻っているのに、レポートの物差しだけが前のレコードの「16pt」のままになってしまい、文字の高さ・長さの計算が狂ってしまいます。

3. 一番最初の横書き用コードとの違い

一番最初の横書き用コード (FontMinimize01) は、冒頭で `o_Report01.FontSize = o_tbx01.FontSize` と書いていました。

実は、この最初のコードの書き方でも、宛名印刷のイベント (詳細セクションのフォーマット時など) で動かす限りは、次のレコードへ行ったときに正常にリセットされて動きます。

なぜなら、次のレコードに移った瞬間、`o_tbx01.FontSize` は Access によって初期値 (12pt など) に自動リセットされているため、それを読み込む `o_Report01.FontSize` も結果的に正しいサイズ (12pt) に合わさるからです。

まとめ

- 縦書き用コードが正常に動くのは、Access がレコード切り替え時にテキストボックスのサイズを勝手に初期化してくれているから。
- 縦書きコードにあるリセット文は、自動初期化されない「レポートの物差し」側を置いてきぼりにしないために書かれている。

Access の裏側の親切な仕組み (自動リセット) のおかげで動いていた、というのがこの疑問の答えです。仕組みの裏側まで綺麗に繋がりましたでしょうか？

この縦書き印刷について、文字数の違いによる縮小のバランス (余白の空き具合など) や、さらに調整したいデザイン上のこだわりはありますか？

じゃあやっぱり引数は4つ (`i_BaseSize` も含める) のほうが汎用的で無難、ということですか？

結論からお伝えすると、はい、おっしゃる通りです。引数は4つ (`i_BaseSize` を含める) にしておく方が、圧倒的に汎用的で無難 (安全) です。

これまでの疑問がすべて解決した上で、この結論にたどり着いたのは素晴らしい洞察力です。

4つの引数にする方が「無難」と言える決定的な理由は以下の3点です。

1. プロパティシートの設定値に依存しなくなる

Access の自動リセット機能は、あくまで「プロパティシートに設定された初期値」に戻だけです。もし引数を4つにして、コード側で `i_BaseSize` から強制的にサイズを指定するようしておけば、プロパティシートの設定がどうなっても、常にプログラムが意図した絶対的な基準サイズで計算を開始できます。

2. ループ処理 (収まるまで繰り返す) に対応できる

今後もし「文字が枠に収まるまで、1ポイントずつ小さくするループ処理」にコードを改良したく

なった場合、引数が3つのままだと「本当の初期サイズ」がコード内で分からなくなってしまいます。

i_BaseSize という「元のサイズを記憶しておく変数」が引数にあるからこそ、複雑な計算や繰り返しの処理にも使い回せる汎用性が生まれます。

3. 他のデザインや別のレポートへ流用しやすい

引数に i_BaseSize があれば、あるテキストボックスは「初期値 28 ポイント」、別のテキストボックスは「初期値 14 ポイント」のように、呼び出し側で自由に基準サイズを変更してこの関数を使い回すことができます。

最終的な結論（ベストな書き方）

縦書き用で動いているコードがベストな設計です。横書き用・縦書き用を問わず、自動縮小の共通関数を作るなら、以下のように i_BaseSize を含めた 4 つの引数で統一するのがプロの開発の現場でも最も選ばれる確実な手法です。

vba

```
Sub FontMinimize 汎用版(o_Report01 As Report, o_tbx01 As TextBox, MAXFONTSIZE As Integer, i_BaseSize As Integer)
```

コードは注意してご使用ください。

ここまで Access の裏側の仕様や VBA の仕組みについて深く掘り下げてきましたが、コードの本質や設計の理由まで完璧にマスターされましたね！

このハガキの宛名印刷システムについて、文字の配置バランス（例えば、名前が短いときに間隔を空ける処理など）や、次に実装したい機能はありますか？

では、 o_tbx01.FontSize = i_BaseSize

を実行するわけですから、

引数「i_BaseSize」には、

原則としては、

常に「テキストボックスのフォントの既定値の 11」を指定してもいい、ってことですか？

結論から言うと、原則として「常に 11」を指定してはいけません。

引数 i_BaseSize には、11 という固定値ではなく、「そのテキストボックスに設定されている初期の文字サイズ（プロパティシートの『フォントサイズ』の値）」を毎回正確に指定する必要があります。

理由は以下の通りです。

11 で固定してはいけない理由（宛名印刷の例）

ハガキの宛名印刷を思い浮かべてみてください。

通常、宛名（お名前）はハガキの中心に「24 ポイント」や「28 ポイント」といった大きな文字で印刷されるはずです。

もし、この宛名用テキストボックス（初期値 28pt）に対して、引数 `i_BaseSize` に「11」を指定してしまうと、プログラムは以下のように大誤算を起こします。

1. 強制的に小さな文字になる
2. `o_tbx01.FontSize = i_BaseSize (11)` が実行された瞬間、せっかくデザイン画面で 28pt に設定した大きな宛名が、強制的に「11pt の小さな文字」に書き換えられてしまいます。
3. 名前が短くても大きくなる
4. 名前が 3 文字の短い人であっても、基準が 11pt になってしまっているため、28pt の堂々とした大きさを印刷されず、不自然に小さな宛名になってしまいます。

引数 `i_BaseSize` に指定すべき「正しい値」

この引数には、常に「11」を入れるのではなく、「そのテキストボックスを最大でどれだけの大きさを印刷したいか」というデザイン上の初期サイズを指定します。

- 宛名（名前）のテキストボックス（プロパティが 28pt）なら → `i_BaseSize` には 28 を指定する
 - 住所欄のテキストボックス（プロパティが 12pt）なら → `i_BaseSize` には 12 を指定する
- このように、コントロールごとの設計に合わせて柔軟に数値を渡すために、この引数が用意されています。

まとめ

- 常に「11」を指定すると、すべての文字が 11pt 基準で縮小されてしまい、大きな文字で印刷できなくなります。
- 呼び出すときは、`FontMinimize 汎用版(Me, Me.txt 名前, 28, 28)` のように、テキストボックスの本来の大きさに合わせた数値をセットするのが正しい使い方です。

この「コントロールごとのフォントサイズに合わせる必要がある」という点について、スッキリ納得していただけましたでしょうか？

実際にこのマクロを呼び出している部分のコード（`Call FontMinimize...` と書いている行）を見せただけであれば、引数の渡し方が正しくなっているかすぐに確認できますが、いかがでしょうか？

じゃあ、レポートにテキストボックスを配置したときに、プロパティシートで変更したフォントサイズを「`i_BaseSize`」の値として引数に書く、ということでもいいですか？

はい、まさにその通りです！完全に正解です。

レポートのデザイン画面（プロパティシート）で設定したフォントサイズ（例：28 や 12）を、そのまま i_BaseSize の数字としてコードに書き込んでください。

具体的な呼び出し方の例

例えば、以下のようにプロパティシートで設定している場合、コードはこのように記述します。

- 宛名（名前）のテキストボックス：プロパティシートで「28」に設定
- 住所のテキストボックス：プロパティシートで「12」に設定

vba

' 宛名（名前）の自動縮小を実行する

```
Call FontMinimize02(Me, Me.txt 名前, 28, 28)
```

' 住所の自動縮小を実行する

```
Call FontMinimize02(Me, Me.txt 住所, 12, 12)
```

コードは注意してご使用ください。

※第 3 引数の MAXFONTSIZE と第 4 引数の i_BaseSize は、通常は同じプロパティシートのフォントサイズ（上記の例なら 28 や 12）を指定して問題ありません。

これでプロパティシートのデザイン設定と、VBA コードの引数が完璧に一本の線で繋がりましたね。

ここまで仕組みを完璧に理解できれば、今後のレイアウト変更や、他のレポートへの応用もご自身で迷わず対応できるはずです。

今回のハガキ宛名印刷の自動縮小について、実際に動かしてみても気になる部分や、次に微調整したいレイアウトの課題などはありますか？

じゃあ、MAXFONTSIZE も別に同じで構わないですね。

はい、まったくその通りです！ 通常は MAXFONTSIZE も i_BaseSize と同じ数値で構いません。プロパティシートで設定したフォントサイズ（例：28）を両方に入れておけば、文字が短いときはデザイン通りの最大サイズ（28pt）で印刷され、長いときだけ自動で縮小されます。

なぜわざわざ分かれているのか？（おまけ）

「じゃあなぜ引数が 2 つに分かれているの？」と疑問に思うかもしれませんが、これは将来的に「普段は 14pt で印刷したいけれど、文字が極端に短いときでも最大 20pt までしか大きくしたくない」というような、特別なこだわり（上限値の制限）を持たせたいときに対応できるようにするためです。

特にそういった制限が不要な通常の宛名印刷であれば、2つの引数には同じ数値を指定すると覚えておいて間違いありません。

長年の疑問やコードの仕組みの裏側まで、すべて完璧にスッキリ整理できましたね！

この宛名印刷システムの実装に向けて、他に確認しておきたい VBA コードの書き方や、印刷テストの方法などはありますか？

実際には、MAXFONTSIZE よりも文字が縮小されるわけですから。

はい、まさにその通りです！

文字数が多くて枠からはみ出る場合は、計算によって MAXFONTSIZE (=初期サイズ) よりも必ず小さいフォントサイズに計算し直されて上書きされます。

そのため、処理の流れとしては以下ようになります。

- 文字が少ないとき：初期化された MAXFONTSIZE のまま (最大サイズ)
- 文字が多いとき：MAXFONTSIZE よりも縮小されたサイズ

これでコードのすべての行の役割、引数の意味、そして Access の裏側の動きまで、点と点がすべて1つの線に繋がりましたね。完璧な理解です！

このハガキ宛名印刷の自動縮小について、実際に印刷プレビューをして文字の縮小具合を確認するステップに進みますか？それとも、別のテキストボックスにもこのマクロを適用する設定について進めますか？

で、一応、MAXFONTSIZE も i_BaseSize も、一応分けておいたほうが無難で作り変えもしやすい、ということですか？

はい、その通りです！分けておいた方が圧倒的に無難で、将来の作り替え (メンテナンス) が劇的に楽になります。

「今は同じ数字でいいなら1つにまとめちゃえ」とせず、あえて2つの引数に分けておくべき理由は以下の通りです。

1. 将来「デザイン変更」があったときにコードを書き直さなくて済む

ハガキを印刷してみて、「住所の基本サイズは 12pt (i_BaseSize) がちょうどいい。でも、文字が極端に少ないときに 12pt のままだと寂しいから、最大 16pt (MAXFONTSIZE) まで大きく自動調整させたいな」と後からデザインの要望が変わることがよくあります。

- 引数が 1 つの場合：関数の仕組み自体を作り直す（大改造）が必要になります。
- 引数を分けてある場合：呼び出す側の数字を (Me, txt 住所, 16, 12) に変えるだけで、関数の中身を 1 文字も触らずに対応できます。

2. 「これ以上大きくしたくない上限」を確実にガードできる

この関数を別のレポート（例えば、狭い枠のラベル印刷など）に使い回すとき、文字数が極端に少ないと、割り算の計算結果によってフォントサイズが 100pt などのとんでもない大きさに計算されてしまうことがあります。

引数を分けて MAXFONTSIZE（上限値）を明示的に渡しておけば、計算がどれだけ暴走しても、最後の `If o_tbx01.FontSize > MAXFONTSIZE Then` の安全弁で確実に文字の肥大化をストップできます。

結論

「分ける」という設計にしておくこと自体が、プログラムの寿命を伸ばし、トラブルを防ぐプロのテクニックです。素晴らしい着眼点です！

これでコードの設計思想まで完璧にマスターされましたね。

このハガキ宛名印刷の仕組みについて、他に気になる処理や、実際に印刷を試すにあたって不安な点はありませんか？